Process platform-based production configuration for mass customization

Zhang, Linda Lianfeng

2007

Zhang, L. L. F. (2007). Process platform-based production configuration for mass customization. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/5527>

<https://doi.org/10.32657/10356/5527>

Nanyang Technological University

Downloaded on 26 Apr 2023 20:55:05 SGT

PROCESS PLATFORM-BASED PRODUCTION CONFIGURATION FOR MASS CUSTOMIZAITON

Linda Lianfeng Zhang

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University

in fulfilment of the requirement for the degree of

Doctor of Philosophy

January 2007

ABSTRACT

A critical issue facing today's manufacturing companies is to achieve product proliferation while maintaining an economy of scale in production. The consequence of high product variety manifests itself through an exponentially increased number of process variations, which introduces significant constraints to production planning and control and prevents make-to-order systems from building up customization capabilities. Managing product differentiation based on product and process families sharing common platforms has been well recognized as an effective means to deal with such a variety dilemma in many industries.

This research proposes systematic methodologies of process platform-based production configuration. The principle is to support production planning in configuring existing operations and processes by exploiting similarity and repetitions inherent in product and process families. Exploiting process families around process platforms helps minimize the variant forms of diverse routings and facilitates the production of a class of products that can widely variegate the operations and process sequences in accordance with specific design changes within a coherent framework.

Mathematical formulation of a process platform is developed to rigorously define process families and the mechanism of production configuration. To model complex data and their relationships underlying production configuration, the structural representation of a process platform is established using the Unified Modeling Language. A systematic methodology based on data mining is developed to construct process platforms by identifying generic routings from historic production information. To support variety coordination from design to production, an approach based on an association rule mining technique is put forward to discover the mapping relationships between product and process variety. To

model the production configuration process, this research develops a set of formalism based on nested colored object-oriented Petri nets with changeable structures.

The proposed concepts and methodologies have been applied to the mass customization of vibration motors in an electronics company. The results of case studies, along with sensitivity analysis, performance evaluation, and system analysis, are discussed in detail, which demonstrate the feasibility and potential of process platform-based production configuration.

ACKNOWLEDGEMENT

I am deeply grateful to my supervisors, Dr. Roger Jiao and Dr. Shaligram Pokharel, for their guidance, encouragement and inspiration throughout the progress of this research and the writing of this dissertation.

Special thanks are due to Dr. Sivakumar, Dr. Arun Kumar, Dr. Yeo Khim Teck, and Dr. Chen Chun-Hsien for their encourage during the course of this research. I would also like to express my appreciation to Mrs. Leong Yuit Lin and Mrs. Tan Lian Hooi for their assistance during my stay in Center for Supply Chain Management (CSCM) and Center for Project Management Advancement (CPMA).

I would also like to acknowledge all the help and inspiration from my fellow classmates: Ms. Lim Ching Moi, Ms. Zhang Yiyang, Mr. You Xiao, and Mr. Ashwin Ravi Ittoo, and other students in CSCM and CPMA. Their cooperation and patience has created a pleasant research environment, where I can focus on my research without trivial troubles.

My gratefulness also goes to my friends, who are studying and working in other places. Without their kind advices, counsel and encouragement, I could not keep the right attitude towards difficulties and challenges encountered during the course of research.

At last, I would like to express my deepest love and appreciation to my dear family. This research would never be possible without their care, love, understanding, and endless support.

TABLE OF CONTENTS

ABSTRACT	I
ACKNOWLEDGEMENT	III
TABLE OF CONTENTS.....	IV
LIST OF FIGURES	VIII
LIST OF TABLES.....	XI
CHAPTER 1: INTRODUCTION	1
1.1. Background.....	1
1.2. Problem Identification	3
1.3. Research Objectives	5
1.4. Research Scope.....	6
1.5. Organization of the Dissertation.....	8
CHAPTER 2: LITERATURE REVIEW	10
2.1. Platform-Based Product Family Design	10
2.1.1. Product Families	10
2.1.2. Product Platforms	12
2.1.3. Product Family Design	13
2.2. Product Configuration	15
2.3. Production Planning in Mass Customization.....	17
2.4. Integrated Product and Process Data Management	19
2.5. Reconfigurable Manufacturing Systems	21
2.6. Summary.....	23
CHAPTER 3: PROCESS PLATFORM-BASED PRODUCTION CONFIGURATION	26
3.1. Concept Implications.....	26
3.2. Principles	28
3.3. Rationale.....	29
3.4. Fundamental Issues.....	29
3.4.1. Process Platform.....	30

3.4.2. Production Configuration	31
3.4.3. Configuration Evaluation	33
3.5. Summary.....	35
CHAPTER 4: FORMULATION OF PROCESS PLATFORM	36
4.1. BOM Models	36
4.2. Fundamental Issues.....	38
4.2.1. Generic Variety Representation.....	40
4.2.2. Generic Routing Structure	41
4.2.3. Generic Planning for Production Configuration.....	42
4.3. Process Platform Formulation	44
4.3.1. Generic Routing Structure	44
4.3.2. Generic Variety Representation.....	48
4.3.3. Generic Planning	53
4.4. Coding-Based Implementation of Variety Coordination.....	60
4.5. Case Study	62
4.5.1. Generic Routing Structure	62
4.5.2. Generic Variety Representation.....	64
4.5.3. Generic Planning	69
4.6. Summary.....	72
CHAPTER 5: STRUCTURAL REPRESENTATION OF PROCESS PLATFORM BASED ON UNIFIED MODELING LANGUAGE.....	73
5.1. Representation Languages	73
5.2. UML Constructs	75
5.2.1. Class Diagram.....	75
5.2.2. Extension Mechanism.....	77
5.3. Structural Representation of Process Platform.....	79
5.3.1. Generic Product Structure	79
5.3.2. Generic Process Structure.....	84
5.3.3. Generic Routing Structure	88
5.4. Case Study	90
5.4.1. Product and Process Information.....	90
5.4.2. Representation of the Motor Process Platform.....	92

5.5. Summary.....	97
CHAPTER 6: GENERIC ROUTING IDENTIFICATION BASED ON DATA MINING	99
6.1. Data Mining Techniques	99
6.1.1. Data Mining.....	99
6.1.2. Text Mining.....	101
6.1.3. Tree Matching.....	102
6.2. Generic Routing Identification	103
6.3. Methodology.....	105
6.3.1. Routing Similarity Measure.....	106
6.3.1.1. Node Content Similarity	106
6.3.1.2. Tree Structure Similarity.....	115
6.3.1.3. Routing Similarity.....	120
6.3.2. Routing Clustering.....	120
6.3.2.1. Principle of Fuzzy Clustering.....	122
6.3.2.2. Procedure of Routing Fuzzy Clustering	124
6.3.3. Routing Unification	126
6.4. Case Study	138
6.4.1. Routing Similarity Measure.....	139
6.4.2. Routing Clustering.....	145
6.4.3. Routing Unification	147
6.5. Performance Evaluation	147
6.6. Summary.....	150
CHAPTER 7: VARIETY COORDINATION BASED ON ASSOCIATION RULE MINING	152
7.1. Association Rule Mining Techniques.....	152
7.2. Association Rule Mining Formulation	155
7.3. Association Rule Mining in Variety Coordination.....	156
7.3.1. Association Rule Mining for Parts	156
7.3.2. Association Rule Mining for Assemblies.....	157
7.3.3. Procedure for Association Rule Mining	158
7.4. Case Study	162
7.5. Sensitivity Analysis	166

7.6. Summary.....	167
CHAPTER 8: PRODUCTION CONFIGURATION MODELING USING NESTED COLORED OBJECT-ORIENTED PETRI NETS WITH CHANGEABLE STRUCTURES.....	169
8.1. Petri Nets for Systems Modeling.....	169
8.2. Production Configuration	172
8.2.1. Production Configuration Process.....	172
8.2.2. Fundamental Issues.....	175
8.2.3. Configuration Description	176
8.3. Modeling Formalism	179
8.3.1. Overview	179
8.3.2. Net Definitions	180
8.3.2.1. Resource Net.....	180
8.3.2.2. Manufacturing Net.....	183
8.3.2.3. Assembly Net.....	185
8.3.2.4. Process Net	186
8.3.2.5. Nested Net System.....	188
8.3.3. Production Configuration Modeling.....	192
8.4. Case Study	193
8.4.1 Production Configuration Modeling.....	193
8.4.2. System Analysis	202
8.5. Summary.....	206
CHAPTER 9: CONCLUSIONS AND FUTURE WORK	208
9.1. Summary of Research Work.....	208
9.2. Conclusions	211
9.3. Contributions	213
9.4. Limitations.....	215
9.5. Avenues for Future Research	216
REFERENCES	218
APPENDIX A: NOMENCLATURE.....	242
APPENDIX B: VIBRATION MOTOR ROUTING DATA	245

LIST OF FIGURES

Figure 1.1 Overview of chapters	9
Figure 2.1 Frame of reference	24
Figure 3.1 Process platform-based production configuration	27
Figure 3.2 Fundamental issues in process platform-based production configuration	29
Figure 4.1 Concept implications of a process platform.....	40
Figure 4.2 Representation of a generic routing structure as a precedence graph	45
Figure 4.3 Classes and their interrelationships within a generic routing structure.....	48
Figure 4.4 Coding-based implementation of variety handlers and their states	61
Figure 4.5 Variety propagation through nesting of variety handlers and of coding.....	61
Figure 4.6 The generic routing structure for vibration motors	63
Figure 4.7 A specific routing for the customized product.....	71
Figure 5.1 Metamodel of UML class diagram	76
Figure 5.2 Layered UML architecture.....	79
Figure 5.3 Links between instances.....	81
Figure 5.4 Representation of the generic product structure.....	84
Figure 5.5 Association between process elements	87
Figure 5.6 Representation of the generic process structure.....	88
Figure 5.7 Association between product and process entities	89
Figure 5.8 Links between product and process entities.....	90
Figure 5.9 Representation of entities and relationships in a process platform	91
Figure 5.10 Vibration motor and its BOM structure	92
Figure 5.11 UML-based representation of the generic product structure of a motor process platform	94
Figure 5.12 A part of the generic process structure of a motor process platform	95

Figure 5.13 Representation of a motor process platform (for Bracket A).....	96
Figure 5.14 A variant of Bracket assy in a particular motor	96
Figure 6.1 Operations and other relevant classes within a generic routing	104
Figure 6.2 A binary tree representation of a routing	105
Figure 6.3 Basic construct of an operation description and bipartite matching	108
Figure 6.4 Component descriptions.....	109
Figure 6.5 An example of tree edit graph.....	118
Figure 6.6 Logic of routing similarity measure.....	121
Figure 6.7 Illustration of fuzzy clustering	124
Figure 6.8 Principle of a generic routing.....	127
Figure 6.9 Procedure of tree growing.....	134
Figure 6.10 Different situations of an l-node arc in the member and generic trees	135
Figure 6.11 Different situations of an i-node arc in the member and generic trees	136
Figure 6.12 Two routing variants for producing vibration motors.....	138
Figure 6.13 Extracted keywords and their occurrences for “ba” component type	140
Figure 6.14 Similarity matrix for primitive component “ba”	141
Figure 6.15 Similarity matrix for compound component “bassy”.....	142
Figure 6.16 Resource similarity matrix for “Bracket Assembly” operation	142
Figure 6.17 Operation similarity matrix for “Bracket Assembly”	143
Figure 6.18 Node content similarity matrix for 30 routings.....	143
Figure 6.19 Tree structure distance matrix of 30 routings	144
Figure 6.20 Tree structure similarity matrix of 30 routings	144
Figure 6.21 Routing similarity matrix	144
Figure 6.22 Routing fuzzy equivalence matrix	145
Figure 6.23 The λ -cut of routing fuzzy equivalence matrix ($\lambda=0.85$)	145
Figure 6.24 Netted graph for the λ -cut in Figure 6.23	146

Figure 6.25 The Generic routing of a motor family	146
Figure 6.26 Sensitivity analysis of routing identification.....	149
Figure 6.27 Performance comparison of the data mining and manual approaches	149
Figure 7.1 Association rule mining in Magnum Opus	165
Figure 7.2 Identified association rules for the part family in Table 7.2	165
Figure 7.3 Generalized generic mapping rules	166
Figure 7.4 Sensitivity analysis of association rules regarding support and confidence levels	167
Figure 8.1 Configuring process elements in production configuration	173
Figure 8.2 Production configuration for a product with an N level hierarchy	174
Figure 8.3 Elements and their connections in production configuration.....	179
Figure 8.4 States and activities of resource objects.....	182
Figure 8.5 Transition decomposition.....	188
Figure 8.6 System model for production configuration modeling	190
Figure 8.7 Communication between port and socket places	192
Figure 8.8 Analogy of decision making for granularity	193
Figure 8.9 Product hierarchies of VM_1 and VM_2	194
Figure 8.10 The $PNet$ of VM_1 and VM_2	196
Figure 8.11 The $MNet$ for part variants Wt_1 and Wt_2	196
Figure 8.12 The $ANet$ for assembly variants MB_1 and MB_2	199
Figure 8.13 Internal behaviors of the assembly machine WcM_1	200
Figure 8.14 Nested net system model of configuring routings for VM_1 and VM_2	201
Figure 8.15 The $MNet$ of part variants Wt_1 and Wt_2	202
Figure 8.15 Procedure of deadlock detection algorithm.....	206
Figure 8.16 A feasible firing sequences leading to the goal marking	206

LIST OF TABLES

Table 1.1 Research objectives and scopes.....	7
Table 4.1 Process variety grid	55
Table 4.2 Operation classes of in the generic routing structure of the vibration motors.....	56
Table 4.3 The operation descriptions associated with each operation class in the generic routing structure of the vibration motors	65
Table 4.4 Variety parameters and their instances defined for vibration motors.....	66
Table 4.5 Variety handlers associated with various generic items within the generic routing structure of the vibration motors.....	68
Table 4.6 Process variety grid for the coil assembly (Aca).....	69
Table 4.7 Specification of a customized product	70
Table 4.8 States of variety handlers relevant to the product	70
Table 4.9 Valid operation variants represented in domains	71
Table 5.1 UML Profile for process platform.....	73
Table 5.2 BOM Data for vibration motor.....	78
Table 5.3 Process data of vibration motor's routing	93
Table 5.4 Description of generic operations for Bracket Assy.....	95
Table 5.5 Specific routing data configured from the process platform in Figure 5.13	97
Table 5.6 The complete routing data for the motor variant in Figure 5.14	97
Table 6.1 Numbers of representation trees for each routing	99
Table 6.2 Attributes and their relative importance	139
Table 6.3 Similarity scales for nominal attributes of “ba”	140
Table 6.4 Result of attribute similarity measure.....	141
Table 6.5 Routing clusters and respective members	146

Table 7.1 Transaction databases of item families.....	158
Table 7.2 Transaction database for a part family	159
Table 7.3 Transaction database for an assembly family.....	164
Table 8.1 Legend in Figure 8.4.....	183
Table 8.2 Legend for Figure 8.6	191
Table 8.3 Specific items of VM_1 and VM_2	191
Table 8.4 Process flows of VM_1 and VM_2	194
Table 8.5 Colored tokens, input and output transform functions for the $PNet$	196
Table 8.6 Process flows of part variants Wt_1 and Wt_2	198
Table 8.7 Resources and operations associated with the $MNet$ and the $ANet$	198
Table 8.8 Colored tokens, input and output transform functions for the $MNet$	198
Table 8.9 Process flows of assembly variants MB_1 and MB_2	199
Table 8.10 Colored tokens, input and output transform functions for the $ANet$	200

CHAPTER 1: INTRODUCTION

This chapter introduces the background and motivation for this research. The research problem is identified as process platform-based production configuration, which means the configuration of production processes or routings for product differentiation by reusing existing process elements and knowledge. The core is a process platform that provides a well-structured mechanism for managing product and process variety within a coherent framework. In accordance with the fundamental issues, the research objectives and scope are defined.

1.1. Background

In today's business environments, mass production of identical products cannot adapt manufacturing companies to the rapid changes in customer requirements and marketplaces. Mass customization has been adopted by many companies in an effort to survive and stay competitive in the current manufacturing reality (Pine, 1993). It aims at the delivery of customized products while achieving mass production efficiency. In mass customization, companies have been striving to design a high variety of customer-specified products (termed product variety; Ulrich and Eppinger, 1995) so as to satisfy diverse individual customer's expectations. Product variety brings competitive edges to companies through offering tailor-made products and more choices to customers (Sanderson and Uzumeri, 1997). To produce product variety, a large number of production processes or routings (referred to as process variety; Jiao and Tseng, 2004) must be planned properly. With reference to the domain framework (Suh, 2001), product variety resides in the design domain and is exhibited by enormous product data recorded in the form of bills of materials (BOMs); while process variety lies in the process domain and is manifested by various routing data.

As evidenced by Wortmann et al. (1997), design specification variations in product variety often lead to enormous and frequent changeovers associated with process variety in production, such as the changes made to operations, process flows and manufacturing resources, including machines, tools and fixtures. In other words, product variety is the source of a myriad of production changeovers on the shop floor. Process variety introduces significant constraints to production planning and control, for example, preventing make-to-order systems from building up customization capabilities (Kolisch, 2000). Today's companies deal with process variety as a type of recurrence problem, that is, to resume production smoothness as soon as possible when facing the unpredictable customer demands. Since a large number of individualized products are often required within a short delivery period, it is difficult for companies to maintain a stable production using the traditional approaches. Companies must develop the ability to produce customized products timely while achieving the economy of scale in production (Jiao et al., 2005a).

Product and process variety become one of the major concerns for companies to deliver customized products quickly with low costs and high quality (Jiao et al., 2007). To achieve this, it is necessary for companies to manage variety of product and process efficiently and cost effectively (Jiao et al., 2005b). With consideration of the causality between the two forms of variety, handling them separately without considering their impact on each other is insufficient for effective variety management. In addition to managing product and process variety individually, variety management should address coordination of variety in the design and process domains. This calls for variety coordination from design to production within a coherent framework (Jiao et al., 2007).

To fulfill heterogeneous customer requirements, the strategy of platform-based product family design involving development of product platforms and design of the associated product families has been adopted by many companies (Simpson, 2004). The efforts are geared towards the realization of design efficiency and effectiveness while satisfying

individualized customer demands. Configuring products from a set of predefined components and/or modules, i.e., product configuration, has drawn much attention from the academia and industry alike (Mittal and Frayman, 1989). Slater (1999) claimed different configurations of a large number of existing components could sufficiently meet diverse customer requirements. In the production field, process planning with respect to computer-aided process planning (CAPP) and computer-aided assembly planning (CAAP) has been intensively investigated for generating automatically detailed process plans for manufacturing parts and forming assemblies. Such detailed plans focus on the specific process parameters, for example, cutting speed, feed rate and orientation of cutting tools. Most of the application systems require inputs regarding manufacturing resources and specifications of parts and assemblies (Erickson, 1988). These inputs are provided in the routings that are planned in advance.

A routing is planned to provide general production instructions for manufacturing a physical product. It consists of a number of ordered operations, the required manufacturing resources, setup activities, and estimated cycle times. In addition to the process flows, routing planning specifies the essential inputs to production, such as parts to be manufactured, assemblies to be formed and resources to be used, as well as detailed process planning for part fabrication and component assembly. Thus, routing planning poses a significant influence on stable production of low costs and short lead times. However, in the manufacturing practice, routings are often planned by planners based on their experiences, intuition, and subjective judgments. No well-structured mechanisms are available for generating routings that can provide more accurate information and data to CAPP and CAAP systems.

1.2. Problem Identification

Most research in product platform development and product family design only encompasses the design domain with emphasis on transforming diverse customer needs to

functional requirements (FRs) and subsequent fulfillment of FRs by a number of design parameters (Simpson, 2004). It seldom, if not at all, explicitly considers the input from the back-end of product realization, i.e., production, in terms of the impact of product platforms on production and, in turn, the implication of process data for product platform development. While design determines the functionalities and performance to be achieved by a product, it is in the production stage that production costs are incurred; product quality and production lead times are determined per se. Therefore, platform-based product family design lacks the support of routing planning.

Current design practice of either developing products based on platforms or configuring products from predefined components and/or modules leads to similarity in product structures, constituent parts and assemblies. In the literature, product similarity and commonality is defined to reflect the extent of similar product elements. In accordance with product similarity and commonality, process similarity and commonality exists in the corresponding routings. It is exhibited by similarity in operations, operations sequences, manufacturing resources, and setups. The full exploitation of similarity and commonality in routings can facilitate the obtainment and maintenance of stable production.

This research puts forward ***process platform-based production configuration*** to deal with the high variety production changeovers in routing planning. It concerns configuring routings for customized products in a family from a process platform of a process family in relation to the product family. The key point in configuring routings lies in the reuse of existing process elements and proven knowledge documented in a process platform while complying with a generic structure, i.e., generic routing structure. Configuring routings from process platforms assists companies in minimizing unnecessary production changeovers caused by routings planned on an ad hoc basis by taking advantages of process similarity and commonality. The reason is that the configured routings are the most similar as the existing ones on the shop floor. The generic routing structure of a process platform entails the

mechanism for effective variety management, which involves managing product and process variety individually and coordinating them using the correspondence in between. This is accomplished by integrating various product and process data into a single entity.

1.3. Research Objectives

Process platform-based production configuration proposed in this dissertation provides a means for companies to minimize changeovers in production by reusing existing process elements and production line setups. In accordance with the fundamental issues identified in process platform-based production configuration (to be introduced in Chapter 3), the research objectives are listed as follows:

(1) *Formulation of a Process Platform.* The first objective addresses process platform-related issues, including concept definitions, functionalities to be provided and possible approaches to achieving functionalities. This is to provide a holistic view of a process platform along with the basic constructs.

(2) *Structural Representation of a Process Platform.* The structural representation of a process platform can facilitate the handling of other issues involved in process platform-based production configuration. The second objective is thus to shed light on process platform structures in terms of the various constituent elements and the complex relationships among them.

(3) *Identification of Generic Routings for Process Families.* The third objective is to address the issues regarding process platform construction. Process platform construction can be facilitated by identifying generic routings of the associated process families because generic routings are fundamental and common to all the specific routings in the associated process family. The rationale of generic routing identification lies in the facts that a large body of product and process data is documented in companies' databases; and the common patterns of routings can be discovered from these historical data.

(4) Identification of Mapping Relationships between Product and Process Variety. The fourth objective is to identify mapping relationships between product and process variety. Product and process variety is exhibited by the various product and process data documented in a process platform. Identifying the mapping relationships between product and process variety thus supports variety coordination from design to production.

(5) Modeling of Production Configuration. The fifth objective is to model production configuration with emphasis on the process of configuring routings from a process platform. It attempts to capture explicitly the fundamental issues involved in production configuration and thus provides configuration automation support.

1.4. Research Scope

Table 1.1 summarizes the more specific research scope in relation to research objectives. To describe rigorously a process platform, the formulation is approached from the architecture point of view, wherein both the static aspect and the dynamic behavior of a process platform are involved. Research is carried out within four areas, including formulation of the generic routing structure that underpins a process platform, formulation of generic variety representation in the generic routing structure, formulation of generic planning for production configuration, and implementation of variety coordination based on coding techniques. In this research, mathematical models incorporating object-oriented (OO) concepts and set theory are established in the formulation.

The structural representation of a process platform involves three aspects. Product items and their relationships are identified and described for modeling the generic product structure contained in a process platform, whilst elements and relationships pertaining to processes are identified and described to represent the generic process structure. The generic routing structure that integrates product and process data is modeled with focus on correspondence

between the generic product and process structures. The Unified Modeling Language (UML) is adopted to establish process platform structural models.

Table 1.1 Research objectives and scope

Research Objectives	Research Areas
Formulation of a process platform	Formulation of the generic routing structure
	Formulation of generic variety representation
	Formulation of generic planning for production configuration
	Implementation of variety coordination
Structural representation of a process platform	Representation of the generic product structure
	Representation of the generic process structure
	Representation of the generic routing structure
Identification of generic routings for process families	Similarity measure of routings
	Clustering of routings
	Unification of routings
Identification of mapping relationships between product and process variety	Mining of specific association rules between product and process data
	Generalization of generic rules for generic items
Modeling of production configuration	Development of a set of modeling formalism based on nested colored object-oriented Petri nets with changeable structures
	Modeling of production configuration based on the developed formalism

Identification of generic routings for process families from historical product and process data is accomplished by three sequential steps, namely similarity measure of routings, clustering of routings and unification of routings. Routing similarity measure compares pairwise the set of given routings. Based on the comparison result, routing clustering is used to group similar routings into families. At last, routing unification forms the generic routings of the associated process families. A systematic data mining methodology is developed in this research to identify generic routings for process families.

Identification of the mapping relationships between product and process variety in a process platform involves (i) uncovering of the associations between specific product and process data and (ii) generalization of the obtained specific rules into generic rules, which correspond with the mapping relationships at the generic level. In this research, an approach based on association rule mining is developed.

Production configuration modeling is performed within two areas, including development of proper modeling formalism and modeling of production configuration by using the developed formalism. Considering the requirements posed by the fundamental issues involved in production configuration, a set of modeling formalism based on nested colored object-oriented Petri nets with changeable structures is developed in this research.

1.5. Organization of the Dissertation

An overview of the chapters in the dissertation is shown in Figure 1.1. Chapter 1 lays the foundation by introducing background, research proposal, objectives, and scope. Chapter 2 gives a review of research work relevant to the proposal, including (i) platform-based product family design, (ii) product configuration, (iii) production planning in mass customization, (iv) integrated product and process data management, and (v) reconfigurable manufacturing systems (RMSs). Besides this general literature review, the literature in relation with the proposed solutions to each research objective is given in the associated chapters. Chapter 3 presents an overview of process platform-based production configuration with respect to concept observations, principles and rationale, and fundamental issues. Chapter 4 reports mathematical models for describing a process platform rigorously. Also discussed is the proposed coding-based implementation of variety coordination in process platforms. A case study of vibration motors for hand phones is given. Chapter 5 provides UML-based structural models of a process platform. A case study of vibration motors is reported to demonstrate UML models of process platforms. Chapter 6 describes a systematic data mining methodology to identify generic routings for process families. The methodology is applied to the same case of vibration motors. Chapter 7 presents an association rule mining approach to discover associations between product and process data included in a process platform. Based on the obtained specific association rules, generic rules are generalized to identify the relations between the generic product and process elements. The case study of

vibration motors is given to demonstrate the feasibility of the rule mining approach. Chapter 8 discusses a set of modeling formalism, which is developed based on nested colored objected-oriented Petri nets (PNs) with changeable structures, and production configuration modeling by using the developed formalism. In the case study, the system model of production configuration is analyzed to prove the feasibility and correctness of the modeling formalism and the constructed system model. Chapter 9 concludes the dissertation by discussing contributions and limitations of the research proposals and the avenues for future research.

Figure 1.1 Overview of chapters

CHAPTER 2:

LITERATURE REVIEW

The background leading to this research encompasses five broad areas, namely platform-based product family design (Section 2.1), product configuration (Section 2.2), production planning in mass customization (Section 2.3), integrated product and process data management (Section 2.4), and reconfigurable manufacturing systems (Section 2.5). In each section, the relevance and limitations of the research work are pointed out to highlight the rationality and significance of the research in this dissertation.

2.1. Platform-Based Product Family Design

Many companies are utilizing the strategy of platform-based product family design to increase variety, shorten lead times and reduce costs in today's highly competitive global marketplace. In platform-based product family design, issues regarding product families and product platforms are concerned.

2.1.1. Product Families

Although researchers define a product family with different wordings, the consensus is that products in a family have common components, features and subsystems, identical main functions, similar product structures, specific features and components, and secondary functionalities (Gonzalez-Zugasti et al., 2000; Farrell and Simpson, 2003; Stadzisz and Henriond, 1995). While the target of a product family is an entire market segment, its product variants (i.e., individualized products) are designed to meet the requirements of each customer in the segment, i.e., segment niches. With product family concepts the challenge of providing customization and variety for the marketplace without losing commonality between customized products can be met (Farrell and Simpson, 2003). Several companies have used

product family design strategy to design successfully their diverse products such as Sony's Walkman product family (Sanderson and Uzumeri, 1997), Nippondenso's bicycles (Whitney, 1993), and Swatch watches (Ulrich and Eppinger, 1995). As pointed out by Stadzisz and Henrioud (1995), a product family may have its origin in a differentiation process of a base product or in an aggregation process of distinct products. In the first case, the family represents a product series with different technologies and optional parts and functions due to the product evolution and demand for diversity. In the second case, the family represents the standardization of a set of products whose functions and main components are similar. In both cases, the goal is to form a group of products to reduce their variability and, therefore, to decrease investment and production costs.

Product families convey different meanings when viewed from different aspects. Du et al. (2002a) observed that in practice, different business functions tended to interpret and employ product families in different ways. From the marketing and sales perspective, product families exhibit the company's product line or product portfolio and thus are characterized by various sets of functional features for diverse customer groups. The engineering view of product families embodies product technologies and associated manufacturability and is thereby characterized by differences in product structures, design parameters and components. A more comprehensive definition of product families from different disciplines, e.g., customer support, project management, system engineering, can be found in van Vuuren and Halman (2001).

As far as the characteristics of a product family is concerned, Collier (1982), McDermott and Stock (1994), Ishii et al. (1995), and Martin and Ishii (1997) identified commonality; Parnas et al. (1985), Sanderson (1991), Ulrich and Tung (1991), Pimmler and Eppinger (1994), Stadzisz and Henrioud (1995), and Martin and Ishii (1997) addressed modularity; Ulrich and Eppinger (1995), Lee and Tang (1997), and Martin and Ishii (1997) emphasized standardization; Rothwell and Gardiner (1990) focused on robust design;

Simpson et al. (1997) related changes in form and function to highlight mutability, modularity and robustness, which they suggested are the core characteristics of product families. Chen et al. (1994) suggested designing flexible product architectures to enable small product changes to increase product variety.

2.1.2. Product Platforms

The key to the success of a product family strategy is the product platform from which the associated product family is derived (Simpson, 2004). A number of perspectives on product platform exist in the literature. A review suggests that product platforms have been defined diversely, ranging from being general and abstract (Corso et al., 1996; Robertson and Ulrich, 1998) to being industry and product specific (Ericsson et al., 1996; Jandourek, 1996; Wheelwright and Clark, 1992; Sanderson and Uzumeri, 1997). In addition, the meaning of platform differs in the scope. Some definitions and descriptions have focused mainly on the product/artifact itself (Meyer and Utterack, 1993; McGrath, 1995), whilst others have tried to explore the platform concept in terms of a firm's value chain (Sawhney, 1998).

There are two streams of product platform research prevailing in the field of developing product platforms to support product family development. One perspective refers to a platform as a collection of elements shared by several related products. Accordingly, the major concern is how to identify common denominators for a range of products (Ericsson et al., 1996; Jandourek, 1996; Kota and Sethuraman, 1998; McAdams et al., 1999). The effort is geared towards the extraction of those common product elements, features and/or subsystems that are stable and well-understood so as to provide a basis for introducing value-added differentiating features (Robertson and Ulrich, 1998; Moore, et al., 1999). Meyer and Lehnerd's (1997) work is the representative of another perspective on product platform. They define a product platform as "a set of subsystems and interfaces developed to form a common structure from which a stream of derivative products can be efficiently developed and

produced”. The major issue is to exploit the shared logic and cohesive architecture underlying a product platform.

2.1.3. Product Family Design

The two basic approaches to platform-based product family design include a top-down (proactive platform) approach and a bottom-up (reactive redesign) approach (Simpson et al., 2001). In the top-down approach, a company strategically manages and develops a family of products based on a product platform and its derivatives, e.g., Sony’s Walkman products (Sanderson and Uzumeri, 1997) and Kodak’s cameras (Wheelwright and Clark, 1995). In the bottom-up approach, a company redesigns or consolidates a group of distinct products to standardize components, for instance, Lutron’s lighting control products (Pessina and Renner, 1998) and Black & Decker’s universal electronic motors (Lehnerd, 1987).

The prominent approach to platform-based product development, be it top-down or bottom-up, is through the development of a module-based product family, wherein product family members are instantiated by adding, substituting and/or removing one or more functional modules from the platform. Some of the more frequently quoted examples are Sony’s Walkmans (Sanderson and Uzumeri, 1997) and Hewlett Packard’s ink jet and laser jet printers (Feitzinger and Lee, 1997). An alternative approach is through the development of a scale-based product family. In a scale-based product family one or more scaling variables are used to “stretch” or “shrink” the platform in one or more dimensions. Some examples include Honda’s “world car” (Naughton et al., 1997), Boeing’s commercial airplanes (Sabbagh, 1996), and Rolls Royce’s aircraft engines (Rothwell and Gardiner, 1990).

An important measure of the success of a product platform is how quickly new products can be developed from it at low costs. To help determine when to renew or refocus product platform efforts, Meyer et al. (1997) introduced metrics for platform efficiency and effectiveness. Platform efficiency assesses the cost of developing a derivative product relative

to the cost of developing the product platform, whilst platform effectiveness measures the ratio of the revenue that a product platform and its derivatives create to the cost required to develop them.

During product platform design, much of the focus revolves around the trade-off between commonality and distinctiveness: designers must balance the commonality of the products in the family with the individual performance (i.e., distinctiveness) of each product in the family. Numerous indices have been developed to measure commonality, e.g., Collier (1981), Baker et al. (1986), Thomas (1992), Lee and Billington (1994), McDermott and Stock (1994), Vakharia et al. (1996), and Kim and Chhajed (2000). The degree of commonality index proposed by Collier (1981) was one of the first such indices that use information contained in the company's BOMs to assess commonality for a single end item, a product family, or an entire product line. Jiao and Tseng (2000) extended Collier's commonality index to create indices for component part commonality and process commonality. Siddique et al. (1998) proposed separate indices for measuring component commonality and connection commonality, applying them to automotive underbodies.

The major emphasis of current practice in developing product families is on the acquisition of customer needs, the subsequent transformation of functional requirements from customer needs, and the final fulfillment of functional requirements by design parameters (Simpson, 2004). Issues regarding production, more specifically routing planning, are almost entirely ignored in the current literature. Only under the condition that routings are planned to the best use of process similarity and commonality resulted from product similarity and commonality inherent in product families, the reuse of existing production line setups and process elements can be obtained to the largest degree. Therefore, a tool needs to be designed to assist routing planning in a systematic way towards the best utilization of process similarity and commonality. Process platforms allow routings to be obtained through configuring existing process elements while complying with a common structure.

2.2. Product Configuration

Product configuration has been an active area of research for the past two decades. It can be applied to a wide range of problem domains including computers (workstations, internetworking, mainframe, and supercomputer hardware etc.), telecommunications systems (e.g., central office switches, and internetworking), transportation (automobiles, trucks and airplanes), modular furniture, custom manufactured materials (e.g., steel), industrial products (e.g., valves, actuators, and controls), and medical systems and services (Franke, 1998). Product configuration assists in (i) improving inter-firm coordination and reducing the trade-off between product variety and delivery time (Forza and Salvador, 2002), (ii) avoiding time-consuming redesign and manual adaptation (Stumptner, 1997), (iii) lowering product development and production costs (Fleischanderl et al., 1998), and (iv) meeting a wide range of customer requirements and increasing control of production (Magro and Torasso, 2003). The two key features of configuration are that (i) the product being configured is assembled from instances of a set of well defined component types, and (ii) components interact with each other in predefined ways (Mittal and Frayman, 1989; Klein et al., 1994; Schreiber et al., 1994; Stumptner, 1997; Franke, 1998; Soinenen et al., 1998; Brown, 1998). Recent work has investigated into the modification of component interface in assembly design (Wang et al., 2005a; b)

Since configuration is synthetic in nature (i.e., the configuration system is expected to produce a new product) (Stumptner, 1997), selecting and arranging parts that satisfy given input specifications form the core of a configuration task. Besides the two subtasks of component selection and arrangement, Brown (1998) also added evaluation of component combination as the third logical subtask. In his evaluation subtask, two tests with respect to component compatibility and goal satisfaction are carried out. During the process of selection and arrangement, no new component types can be created, and the interface of the existing

component types cannot be modified. The solution must produce the list of selected components (i.e., part lists of both systems and individual components) as well as the product's structure and topology (Sabin and Weigel, 1998).

While the input specifications may be incomplete, ambiguous, incrementally evolving, granular to different levels of specificity, inconsistent, entered in any arbitrary order, interconnected and nested with complex structures; the output must be complete, consistent (with respect to other parts, preferences, pre-existing components), modifiable, understandable/explainable, interconnected and interoperable with related data (Franke, 1998).

Two main characterizations of product configuration formalization are (i) describing the configuration in terms of constraint satisfaction problems (CSPs; e.g., Mittal and Falkenhainer, 1990; Stumptner and Haselbock, 1993; Sabin and Freuder, 1996; Fleischanderl et al., 1998; Soininen and Gelle, 1999), and (ii) characterizing configuration in terms of logical approaches (e.g., McGuinness and Wright, 1998; Friedrich and Stumptner, 1999; Soininen et al., 2000). While logical approaches to configuration stress the need for an explicit representation of the structural properties of the entities of the domain in terms of specific kinds of relations, CSPs look at configuring products as solving a problem with a number of restrictions (Magro and Torasso, 2003). Some authors also mentioned the importance in representing the knowledge on a configuration and the restrictions on possible configurations (Soininen et al., 1998; McGuinness and Wright, 1998).

Following R1/XCON for configuring computers at Digital Equipment Corporation (McDermott, 1982), numerous configuration systems have been developed by other organizations. In accordance with the configuration reasoning logic, most of these systems can be classified into three types: (i) rule-based reasoning, (ii) model-based reasoning, and (iii) case-based reasoning (Sabin and Weigel, 1998). With incorporation of the advanced Internet technologies, a number of Web-based configuration systems (called configurators)

have been developed and are in use, e.g., Pconfig (Slater, 1999) and CAWICOMS (Ardissono and Goy, 2000). These configurators are expected to enable businesses to market complex customizable products and services by using the new technologies of electronic commerce, whereby customers can tailor the configurable products according to their specific needs and requirements.

The basic principle of product configuration, i.e., selecting elements from existing ones and accordingly arranging the selected elements, lays foundation for process platform-based production configuration. The difference is that in process platform-based production configuration, the efforts are made to select existing compatible process elements and organize them into routings.

2.3. Production Planning in Mass Customization

Production planning has attracted a lot of attention and interest from academia and industry alike ever since the identical products in mass production could not satisfy the heterogeneous customer requirements in the business markets. The enormous research efforts in production planning have been dedicated to the determination of master production schedules and material requirement planning (Turner et al., 1992). In this regard, production planning takes routings of products for granted. As downstream activities of production planning, CAPP, and CAAP aim to generate detailed process plans for producing parts and forming assemblies, respectively.

CAPP plays an important role in linking design and manufacturing activities (Ming et al., 1999). At the very beginning, the development of CAPP only attempted to build computer assisted systems for report generation, storage and retrieval of plans (Chang and Wysk, 1985). Only recently, CAPP systems have been developed by focusing on the entire process planning activities, including finding the relevant processes for manufacturing a part, sequencing these processes and specifying the complete set of process parameters for each

process (Schierholt, 2001). In addition to two traditional approaches to CAPP, i.e., variant and generative approaches, Ming et al. (1999) also discussed another two inference methods in CAPP systems, including expert system based inference and neural networks based inference. While inputs to CAPP systems vary widely, the general types of inputs to most systems include product data, resource data and decision logic (Erickson, 1988). Both product and resource data are specified by routings of the complete products.

Assembly planning tries to determine a detailed plan to construct an assembly from its component parts. Due to the complexity and intrinsic combinatorial explosion of the problem to be solved and the variety of assembly topologies, automation and computerization of assembly planning, i.e., CAAP, is a critical area in factory integration and automation (Groppetti et al., 1994). Systematic assembly plan generation in CAAP can ensure that no good sequences are overlooked, provide a means to evaluate alternatives which better utilize resources in a flexible environment, and assist designers in assessing different design solutions for a given product based on assembly process requirements (Homem de Mello and Sanderson, 1991).

CAAP has received much attention over the past decade. A literature review suggests most research in CAAP focuses on four aspects, namely assembly modeling (e.g., Lee and Shin, 1990; Pham and Dimov, 1999; Wang and Kim, 1999), assembly plan representation (e.g., Gottipolu and Ghosh, 1997; Qian et al., 1996; Homem de Mello and Sanderson, 1990), assembly plan generation (e.g., Rohrdanz et al., 1996; Chakrabarty and Wolter, 1997; Gottipolu and Ghosh, 1997), and assembly plan evaluation (e.g., Lee, 1992; Wolter and Chandrasekaran, 1991).

As the outputs of numerous research efforts, a lot of CAPP and CAAP systems or prototypes have been developed. Some examples of CAPP systems are APPAS (Automated Process Planning and Selection; Wysk, 1977), ACAPS (Automated Coding and Process Planning Selection; Emerson and Ham, 1982), CMPP (Computer Managed Process Planning,

Waldman, 1983), and PROPLAN (Phillips and Mouleeswaran, 1985). Two examples of CAAP systems are HighLAP (Rohrdanz et al., 1996) and MAPS (Manufacturing Assembly Process Sequencer; Gao and Bowland, 2002).

Whatever the research efforts devoted in CAPP and CAAP, they target part manufacturing or assembly formation rather than production of complete products. Producing complete products requires both manufacturing and assembly operations that are dealt with separately by CAPP and CAAP, respectively. Further, their application requires product (either parts or assemblies) data and resource data as inputs, which are provided by routings of the complete products. Consequently, the accuracy of their inputs can only be guaranteed by routings. Therefore, only such routings that are most similar to the existing ones on the shop floor can ensure the usefulness of the generated detailed manufacturing and assembly process plans. Process platform-based production configuration can help companies to obtain such optimal routings. Moreover, incorporating routings configured from process platforms, companies can determine more realistic master production schedules and material requirement plans.

2.4. Integrated Product and Process Data Management

The fast delivery and low cost production of high quality customer-specified products puts design and production integration into many research agendas. The intent of integrating design and applications following design (e.g., planning, scheduling) is to incorporate production considerations into product design phase to evaluate the manufacturability and the production cost of the design (Stadzisz and Henrioud, 1995). The benefits of integration ranges from lowering product development costs, reducing time-to-market, improving the product quality, increasing the communication among different business functions to better utilizing manufacturing resources. The linchpin in design and production integration lies in the integrated product and process data management (Lee and Chen, 1996). Integrated

product and process data management helps design and production engineers manage both data and product development process. It keeps track of the masses of data, information and knowledge required to design and produce products (Philpotts, 1996). In other words, integrated product and process data management is central to design and production integration (Wu and Wu, 1998; Kovacs et al., 1999).

Not only has the importance of dealing with the coupling between design and production been well realized, but also design and production integration based on integrated data management has motivated a myriad of approaches to create a common working platform for design engineers and production system developers, such as the work reported at the CE-NET (<http://www.ce-net.org/>) and the assembly net (<http://www.assembly-net.org/>). In this regard, the interest in integrated product and process models are evident from the research agendas and improvement programs that have emerged. Svensson (2001) emphasized the proper handling of customer and production data based on BOM information to achieve an alignment of products and processes. The review by Huang et al. (2003) suggested that product structures and cost data, along with proper production information models, impose a significant impact on production planning throughout a manufacturing supply chain. Lea and Fredendall (2002) studied product mix decisions *vis-à-vis* different product BOM and routing structures in a shop floor. Kovacs et al. (1998) discussed collaborative product and process design by integrating product data management (PDM) systems and workflow management systems (WfMS). While PDM systems rely on product breakdown structures (PBS), WfMS facilitate production information management through mapping PBS to work breakdown structures (WBS) or assembly breakdown structures (ABS).

Roucoulès et al. (2003) discussed integration of process planning knowledge in the two phases of product design. They described process planning as part of the detailed design phase (i.e., the second phase), which requires the minimal structural data from the first phase, the initial design phase. Zha (2000) defined a new unified class of object-oriented knowledge

based Petri nets an approach to integrating assembly design and planning. Zha and Du (2002) presented a STEP (Standard for Exchange of Product model data) -based method and system for integrating assembly design and process planning. The integrated prototype with the embedded STEP-based generic product assembly model was developed to support the introduction of new products. Intelligent agents were used to carry out the integration of preliminary design and manufacturing planning (Feng and Song, 2002).

Along this stream, researchers have concentrated their efforts on the development of frameworks, systems and techniques that embody the key characteristics and dimensions relevant to optimization of design and production of individual products. In the context of mass customization, however, the focus is on the mass production efficiency of the cohort, instead of individuals, of diverse product variants and product families, where an individual may not be optimal. Therefore, these traditional integrated design and production models along with the underlying integrated product and process data cannot adapt themselves to satisfy current product development requirements. In contrast, process platform-based production configuration attempts to address issues regarding mass production efficiency of product families. It is accomplished by integrating product and the associated process data into a unified structure.

2.5. Reconfigurable Manufacturing Systems

Today's unpredictable market changes occurring at an increasing pace have put RMSs in the research agenda (Koren and Ulsoy, 1997; Koren et al., 1999). The emergence of RMSs is to overcome the limitations of traditional manufacturing systems (e.g., dedicated manufacturing systems) and conventional manufacturing systems (e.g., cellular manufacturing systems, flexible manufacturing systems) by accommodating customer required product variety, manufacturers' desired productivity and production costs (Koren and Ulsoy, 1997; Koren et al., 1999; Zhao et al., 2000).

The research group at the NSF Engineering Research Center for Reconfigurable Manufacturing Systems (ERC/RMS) at The University of Michigan described the goal of an RMS as for rapid adjustment of production capacity and functionality in response to new circumstances by rearrangement or change of its components (Koren and Ulsoy, 1997; Koren et al., 1999). Components may be machines and conveyors for entire production systems, mechanisms for individual machines, new sensors, and new controller algorithms. New circumstances may be changing product demand, producing a new product on an existing system, or integrating new process technologies into existing manufacturing systems. Other authors, e.g., Zhao et al. (2000) and Abdi and Labib (2003), extended this definition to multiple product families. In their definition, products are not limited only to parts. They may be complex products that consist of both parts and assemblies. In spite of the difference of targeted products, the objective of RMSs remains the same as to provide exactly the functionality and capacity that is needed, exactly when it is needed (Koren and Ulsoy, 1997). The key characteristics of RMSs are modularity (design all system components, both software and hardware, to be modular), integrability (design systems and components for both ready integration and future introduction of new technologies), convertibility (allow quick changeover between existing products and quick system adaptability for future products), diagnosability (identify quickly the sources of quality and reliability problems that occur in large systems), and customization (design the system capability and flexibility (hardware and controls) to match the product families) (Mehrabi et al., 2000).

A number of researchers have investigated RMSs from various aspects with focus on different issues. The researchers at the NSF ERC/RMS approach RMSs in four trust areas (TAs), including TA-1: system level design, TA-2: controls, TA-3: inspection and ramp-up, and TA-4: responsive maintenance (<http://erc.engin.umich.edu/research.htm>). Each TA has its final research goal and several research objectives that can fulfill the goal. In addition to the work at the ERC/RMS, other authors have also done quite a lot of research in RMSs.

Zhao et al. (2000) described a framework of RMSs considering three important issues, namely optimal configurations in RMS design, optimal selection of product families and the corresponding configuration of manufacturing systems, and performance measure of the configured systems. Abdi and Labib (2003; 2004) proposed a strategy for designing RMSs and grouping products into families by using the analytical hierarchical process technique. Chick et al. (2000) presented a descriptive multi-attribute model of reconfigurable machining system selection process with consideration of buyer and supplier relationships. Zhang et al. (2002) discussed a multi-agent based architecture for supporting the design and implementation of highly reconfigurable control systems for manufacturing cells. A knowledge-based computer-aided configuration tool was developed to enable easier and faster manufacturing system reconfiguration (Tonshoff and Drabow, 2003).

The review suggests two research directions in RMSs. Research along the first direction focuses on the handling of reconfiguration issues for part manufacturing, e.g., the work in the ERC/RMS. As a result of considering only part manufacturing, the proposed methods and frameworks for reconfiguring manufacturing systems cannot work well for products that include both parts and assemblies. In the second direction, researchers from other universities, e.g., Zhao et al. (2000), address the handling of manufacturing system reconfiguration for different product families with emphasis on the production changeovers among product families. Consequently, they did not address production variations of products belonging to the same families. As a contrast to the research in RMSs, process platform-based production configuration tackles production variations both within and between product families. In other words, it overcomes the limitations of RMSs.

2.6. Summary

This chapter reviews the many approaches, methodologies and systems that are developed to help companies implement a mass customization strategy successfully. Besides

the benefits and advantages, the relevance, deficiency and limitations of current research work are also pointed out. In turn, the unaddressed issues and particular aspects of problems in the reviewed research provide opportunities for proposing process platform-based production configuration in current high variety-involved manufacturing environments. Research areas reviewed include platform-based product family design (product family concepts, product platform concepts and product family design), product configuration, production planning in mass customization (more specifically CAPP and CAAP), integrated product and process data management, and reconfigurable manufacturing systems. Figure 2.1 shows the associations between process platform-based production configuration and the relevant literature.

Figure 2.1 Frame of reference

Similar to the concept of product platforms for product families, process platforms are put forward for process families of the associated product families. In other words, a process platform of a process family corresponds to the associated product family and its platform. Thus, the routings to be configured from a process platform is to produce a particular product in the corresponding product family. The limited consideration of routing planning in product family design based on product platforms motivates the proposal of configuring routings from process platforms. The principles of production configuration are borrowed from that of product configuration. The focus on individual products in traditional design and production integration based on integrated product and process data management also provides the

opportunity for process platform-based production configuration. Similarly, limitations of the research in RMSs provide the opportunity for the proposed research. The usefulness and efficacy of the process plans generated by CAPP and CAAP as well as the practicability of master production schedules and material requirement plans are determined by routings in that CAPP, CAAP and production planning require routings to provide inputs.

CHAPTER 3:

PROCESS PLATFORM-BASED PRODUCTION CONFIGURATION

This chapter presents an overview of process platform-based production configuration. Its position in mass customization is recognized and its concept implications are highlighted as well. The principles and rationale of process platform-based production configuration are explained in detail. The fundamental issues in process platform-based production configuration are identified. In accordance with the fundamental issues, the possible technical challenges that may be encountered are discussed.

3.1. Concept Implications

Current research addressing new product development has explored product similarity and commonality, which leads to design efficiency and effectiveness. However, investigations tackling the use of product similarity and commonality for realizing the economy of scale in production are still scarce. Therefore, this research proposes process platform-based production configuration for companies to configure routings for customized production in a family from a process platform in relation to the product family. Such configured routings possess similarity in process elements such as operations, operations sequences, and manufacturing resources. As a result, frequent and unnecessary changes in production caused by routings planned on an ad-hoc basis are expected to be eliminated. Accordingly, stable production for a high variety of customized products is expected to be obtained and maintained.

Apart from accommodating routing planning by providing a generic routing structure and a set of configuration rules, a process platform is able to address coordination of product and process variety as shown in Figure 3.1. As pointed out by Jiao et al. (2007), variety

coordination from design to production is the major concern in mass customization. Therefore, a process platform is proposed in a way that can assist companies in achieving effective variety management and routing configuration. In other words, the basic mechanisms in process platforms for variety coordination support the configuration of existing process elements to generate the routings that are most similar to the existing ones on the shop floor.

Figure 3.1 Process platform-based production configuration

Implications of process-platform based production configuration are summarized as follows:

- (1) Production configuration is conducted within a process platform associated with a product family. In other words, a process platform must be available for production configuration.
- (2) The results of production configuration are routings for customized products of the associated product family. Each routing contains the entire set of operations necessary for producing the given product, the required manufacturing resources for each operation, the estimated cycle times, and setups.
- (3) Production configuration is carried out for a given product, that is, the inputs of production configuration are product specifications.
- (4) Essential to production configuration is selection of the process elements and arrangement of the selected process elements. Considering the relationships among multiple process

elements, elements arrangement includes: (i) connecting operations in the correct order, and (ii) associating manufacturing resources, cycle times, and setups with the correct operations. (5) Both the selection and arrangement of process elements are required to satisfy certain constraints or configuration rules embedded in process platforms. (6) The configuration rules should specify the relationships between items of a given product and the relevant process elements as well as the relationships among process elements.

3.2. Principles

Originating from the idea of product configuration, production configuration incorporates the principles of product configuration into that of routing planning. Traditionally, routing planning adopts the strategy of problem decomposition and solution aggregation attempting to simplify the complex planning problem. Before planning routings for given products, the product is broken down into several easily-manageable sub-systems at different levels of abstraction. Each sub-system is treated as a final product when planning its routing. The routing for the given product is formed by aggregating the routings of the sub-systems according to the relationships among these sub-systems. Product decomposition is performed by following the hierarchical structure of the given product. The basic principle of product configuration is to select and arrange compatible components/modules while meeting several restrictions from the set of predefined ones in a configuration system. Therefore, the principles of production configuration can be summarized as: (1) decompose the given product according to its hierarchical product structure; (2) select process elements from the process platform for each decomposed items and connect the selected elements into routings while satisfying configuration rules; and (3) form the routing for the given product by aggregating the routings for decomposed items according to their relationships.

3.3. Rationale

Due to product similarity and commonality, routings for producing products in a family share a number of identical and similar operations, manufacturing resources and operations sequences. Furthermore, the common product structure assumed by a family of customized products entails a common process structure shared by all routings in the associated process family. A process platform of a process family in relation to a product family can thus be built around the common process structure. Integrating all product data and corresponding process data into the common process structure using a generic representation (to be introduced in Chapter 4) results in the generic routing structure that underpins the constructed process platform. To summarize, the rationale of production configuration is process similarity and commonality, which is reflected by similar operations, operations sequences and manufacturing resources utilized in different routings.

3.4. Fundamental Issues

Figure 3.2 illustrates the fundamental issues involved in process platform-based production configuration. Three areas include: (i) process platform, (ii) production configuration, and (iii) configuration evaluation. A number of specific research issues in each area are identified. Some technical challenges may be encountered when addressing these issues as discussed in the following sections.

Figure 3.2 Fundamental issues in process platform-based production configuration

3.4.1. Process Platform

The first issue regarding process platforms deals with the rigorous description of a process platform in terms of its concepts and functionality. This issue is addressed by the first research objective in this dissertation: formulation of a process platform. Process platform formulation requires the establishment of such models that can lead to a holistic view of a process platform along with the basic constructs. Therefore, the technical difficulties in formulating a process platform includes (i) building process platform models, (ii) introducing logically other concepts relevant to the models, and (iii) stating accurately the functionality in a way that can facilitate computer implementation. To satisfy the basic formulation requirement and to meet the technical challenges, the OO concepts and set theory are adopted in this research to build mathematical models of a process platform.

The second issue focuses on the constituent elements of a process platform. This issue is tackled by the second research objective: structural representation of a process platform. The models representing a process platform structure must have the ability to describe explicitly and accurately the various entities and the complex relationships among them so as to facilitate future development through model analysis. In accordance with the representation requirements, the technical challenges are observed as (i) representation of generic structures, (ii) representation of relationships within generic structures, and (iii) representation of relationships between generic structures. This research adopts UML as the representation tool in order to handle the technical challenges and build the process platform structural models accurately.

The third issue relates to process platform construction. The third research objective in this dissertation: identification of generic routings of process families is proposed to support construction of a process platform. The suggested solution approaches process platform construction as identification of the generic routings for process families from historical

product and process data. The technical challenges in generic routing identification lie in: (i) measuring similarity among given routings, (ii) grouping routings into families, and (iii) forming the generic routings of process families. To meet the challenges, the proposed solution adopts data mining techniques, including text mining and tree matching, for measuring routing similarity, fuzzy clustering and netting graph method for forming process families, and tree unification for obtaining generic routings.

The fourth issue emphasizes coordination of product and process variety in process platforms. This issue is addressed by the fourth research objective in this dissertation: identification of mapping relationships between product and process variety. The technical difficulties exist in the obtainment of the associations between massive product and process data. The proposed solution adopts the association rule mining technique to discover common associations between product and process data, which embody the mapping relationships between product and process variety.

3.4.2. Production Configuration

Production configuration issues include: (i) production configuration formulation, (ii) the production configuration space modeling for configuration reasoning, and (iii) production configuration modeling with emphasis on the configuration process.

Production configuration formulation aims to state unambiguously the problem nature of production configuration. Production configuration involves the selection and arrangement of finite process elements for a given product while satisfying a number of configuration rules. Process elements can be treated as a set of variables, whose value determination depends on the given input and configuration rules, configuration rules can be dealt with as a set of constraints; and the given input can also be handled as a set of variables, each of which has a finite set of values. Hence, production configuration by nature can be treated as a constraint satisfaction problem and formulated accordingly. As stated by Bain et al. (2004),

the paradigm of CSP has been accepted as a popular one for representing finite domain problems. The technical challenges in formulating production configuration as a constraint satisfaction problem may involve: (i) representing process elements as a set of problem variables, (ii) representing domain values for problem variables, (iii) representing product specifications as a set of input variables, (iv) representing domain values for input variables, (v) representing configuration rules as a set of constraints, and finally (vi) applying constraints (configuration rules) to problem and input variables (process elements and product specifications) to obtain domain values for problem variables.

The second issue concerns production configuration space modeling for configuration reasoning. Similar to many engineering design problems, the problem of production configuration can be tackled as search in a space, wherein each space element represents a combination of process elements, i.e., a configuration solution, may it be feasible or infeasible. With the introduction of constraints, the search will be carried out in the feasible regions (formed by feasible configurations) only, and the infeasible regions will be eliminated from consideration. Therefore, the computational effort in searching possible solutions will be reduced dramatically by applying these constraints. Different from the spaces of engineering design problems, the production configuration space is discrete and combinatorial (but not purely combinatorial) in nature. This is because the configured process elements are not real-valued, and further, the configuration results are represented by discrete structures, for example, graphs. As pointed out by Siddique and Rosen (2001), proper modeling of design spaces can greatly ease the search for solutions. This is especially true for the production configuration problem. The reason is that the production configuration space is more complex than the design space, which is not combinatorial and is characterized by real-valued elements. Therefore, it is imperative to represent the configuration space for production configuration reasoning. Some technical challenges may be encountered during the course of modeling, which include: (i) representation of the combinatorial configuration

space, (ii) modeling of constraints and their effects on the space, (iii) representation of feasible regions and efficiently enumerate configuration solutions within the feasible region, and (iv) determination of the size of feasible regions.

The identified issue relevant to production configuration is to model production configuration formally with emphasis on the configuration process. The last research objective in this dissertation: production configuration modeling, addresses this issue. In light of the fundamental issues in production configuration (discussed in Chapter 8), the technical difficulties in modeling production configuration involve: (i) representing the large number of process elements of both same and different types, (ii) handling constraints for the selection and arrangement of process elements, (iii) modeling configuration at different granularity levels, (iv) representing structural changes in routings being configured, and finally (v) modeling communication between configuration models at different levels. For meeting these technical difficulties, this research develops a set of modeling formalism based on nested colored object-oriented PNs with changeable structures and subsequently applies the developed formalism to build the production configuration models.

3.4.3. Configuration Evaluation

Evaluation attempts to obtain an optimal result among all the possible ones of the same solution. The reason for one solution generating more than one result is many-fold. For example, different parameter settings in the same solution may produce different outputs, or due to the nature of the problem multiple results can satisfy the exact same requirements set in the solution. In fact, evaluation in process platform-based production configuration needs to handle both situations. It attempts to obtain such process platforms and configured routings that can contribute significantly to the success of mass customization. In this regards, two kinds of evaluation are involved: (i) the evaluation of process platforms, and (ii) the evaluation of configured routings. Process platform evaluation should guarantee that the

obtained process platform possesses the generic routing structure, which is the most common to all individual routings. It is not necessary that there be a one-to-one correspondence between process platforms and product families. The evaluation may produce two process platforms for a product family, which depends on how common and similar the product variants in the family are. Evaluation of configured routings involves two aspects. First, the evaluation is conducted among a number of routing alternatives that are generated for a single product. Among them, an optimal one can be specified. However, the optimal routing for each individual product may not be the optimal one when considering the cohort of a product family. Thus, the second aspect deals with the evaluation of all generated routings with consideration of all customized products in the family as a whole. The optimal set of routings with respect to each associated product can be determined in this way.

Although process platform and routing evaluation targets different objects, it is not necessary to perform them separately. They may be achieved in either two stages or one solution approach or in one stage. In the literature regarding evaluation of product platforms and the associated product families, some authors proposed separate evaluation for product platforms and product families to be designed. In other words, product platform evaluation and product family evaluation are conducted at two stages (Fujita et al., 1999; Simpson et al., 1999; Gonzalez-Zugasti et al., 2000). Some authors introduced one stage approaches to evaluate the desired product platforms and the family of products that can be designed from the platforms (Farrell and Simpson, 2003; Li and Azarm, 2002; Fellini et al., 2000). They used the same objective functions to derive both the optimal product platforms and the families to be designed. Production configuration can employ the same principles of product platform and product family evaluation, that is, both process platforms and configured routings can be evaluated in one stage simultaneously, or the evaluation of process platforms in the first stage and the evaluation of configured routings from the evaluated process platforms in the second stage. In any case, the technical challenges may lie in: (i) selection of

performance measures, (ii) selection of objectives and the formulation of the objective functions, (iii) determination of decision variables, and finally (iv) selection of the problem solving techniques to solve the objective functions.

Finally, the three pillars, as shown in Figure 3.1, are not isolated with one another but are correlating in a certain way. Configuration evaluation depends on the solution approaches to constructing a process platform and the configured routings. The constructed process platforms affect the routings to be configured in terms of its process elements and the structures, which in turn influence the degree of the economy of scale in production that can be realized.

3.5. Summary

In this chapter, process platform-based production configuration is discussed from several perspectives. First, the position in mass customization and the concept implications are presented. Subsequently, the principles and the rationale of process platform-based production configuration are analyzed. Principles of production configuration are formed by combining these of product configuration and routing planning, and thus include product decomposition, process element selection and routing formation. The rationale lies in process similarity and commonality resulting from product similarity and commonality. The fundamental issues in process platform-based production configuration are discussed along with the possible technical challenges. Three areas, including process platform, production configuration, and configuration evaluation, are identified as the basis for of process platform-based production configuration. More specific issues in each area are identified. In the next chapter, the basic issues of process platforms in terms of concepts, functionality and approaches to realizing functionality.

CHAPTER 4:

FORMULATION OF PROCESS PLATFORM

The fundamental issues of process platforms are identified as generic variety representation, generic structures and generic planning for production configuration. Process platform formulation is approached from the three fundamentals accordingly. Coding techniques in group technology are adopted in the proposed implementation of variety coordination. All data contained in product and associated process families are organized in a way that eases the configuration of routings. Therefore, the data structures of process platforms resemble that of the unified BOM and production processes. In the next section, BOM models (including integrated BOM models) are reviewed.

4.1. BOM Models

In traditional methods, product and process planning activities relate directly to product structures in the form of a BOM (Stonebraker, 1996). Though many names and definitional distinctions appear in the literature, common types of BOM construction include modular BOM (MBOM), percentage BOM (PBOM), super BOM (SBOM), variant BOM (VBOM), and generic BOM (GBOM).

While the traditional BOM links components to end items, an MBOM links components to product options, which is below the end product level (van Veen and Wortmann, 1992). The modularization does reduce the number of bills by segregating common parts and disentangling product feature combinations (Stonebraker, 1996). However, the problem of an enormous number of BOMs can only partly be solved by modularizing BOMs due to the formidable hindrances underlying the MBOM method as observed by Jiao et al. (2000). A PBOM captures the proportional composition of either parts or modules so as to support the translation of the forecast volume for the product family as a whole into the volumes for

components. Since the forecast and planning of aggregate is generally more accurate than for components, the PBOM may enhance product forecast and scheduling efficiency (Mather, 1987). An SBOM augments MBOM and PBOM with add/delete bills as attachments (Kneppelt, 1984). Minor product variations are handled by a simple add/delete attachment representing only the change, thus avoiding the creation of a complete new BOM. The VBOM approach is to construct a specific multi-level BOM by selecting none or one BOM relationship variant from each cluster in the multi-level VBOM (van Veen, 1992). The VBOM excels in generative BOM processing especially for those situations where product variety arises at the top-levels of product structures. To overcome the limitations of the VBOM as observed by van Veen and Wortmann (1992) in variant handling, the GBOM concept has been developed (Hegge and Wortmann, 1991; Hegge, 1992; van Veen, 1992; van Veen and Wortmann, 1992; Callahan, 2006). The GBOM empowers the specification of product variants by means of an item and a set description at any level in a multilevel BOM rather than only top-level items. The GBOM excels in describing a large number of variants within a product family in the context of product structures, which is engineering-oriented. Since the variety dilemma involves both product engineering and production management, the consequence of variety's propagation to production deserves further exploration of the GBOM concept in terms of dealing with various routing variations and operations changeovers.

With regard to integrated production data modeling, Hastings and Yeh (1992) proposed a concept of bill of manufacture (BOMfr). The BOMfr combines the routing with traditional BOMs to provide material requirements data for each operation scheduled, resulting in a time-phased material requirement plan derived from a feasible schedule. Blackburn (1985) demonstrated how combining routings and BOMs in one document can support just-in-time manufacturing in traditional Material Requirements Planning (MRP) implemented production environment such as job-shops. Jiao et al. (2000) put forward a generic bill-of-materials-and-

operations (GBOMO) for the purpose of unifying BOMs and routings as well as accommodating large numbers of product and process variants. Tatsiopoulos (1996) discussed the consequences of unifying BOMs and routings on the basic functions of production planning and control. Ou-Yang and Pei (1999) developed an integrated environment to link design tasks with MRP activities. For each engineering change activity, the engineering bill of material data about the changed design stored in a computer-aided design database is extracted and transformed to manufacturing bill of material data stored in the MRP database.

4.2. Fundamental Issues

The general idea of platform thinking (Sawhney, 1998) is to develop product families and the associated process families together so as to produce high variety while maintaining the economy of scale in production. In relation to a product family, a process family refers to a set of similar routings that are planned to fulfill all individual products belonging to the family. Similar to terming an individual product in the family as a product variant, its routing in the process family is called a routing variant. Propagated from product commonality and similarity, commonality and similarity exist in routing variants in the process family with respect to the product family as mentioned earlier. In mass customization, all variants of a product family, while possessing certain differences, are assumed to conform to a similar or identical product structure and follow a similar or identical process flow (Wortmann et al., 1997). This means there exist a common product structure and a common process structure within a product family, and variety is embodied by different variants (instances) of these common structures.

While a product family can be developed from a product platform, the routings of the product family (i.e., the process family) can be configured from a process platform. This research defines a process platform as *a set of common and selective process elements*

corresponding to the set of common and optional product items, a set of relationships among the process elements, a set of relationships among the product items and process elements, and finally a set of constraints specifying the activation of these relationships. Process elements include operations, manufacturing resources (e.g., machines, tools, fixtures, jigs, etc), cycle times, and setups. Product data include data such as part data, assembly data, and design parameters.

A process platform involves *de facto* three aspects: (i) a common routing structure shared by all routing variants, (ii) configuration of specific routings from the common structure, and (iii) correspondence between product and process data, which resembles the correlation between the product and process variety. In this research, the three issues are approached by generic structures, generic planning in production configuration and variety parameters, respectively.

Figure 4.1 illustrates the process platform concept. As shown, each generic or specific process (note, the process here means the one for producing one product item rather than the complete routing for the end product), may it be a manufacturing type or an assembly one, contains one or more than one ordered operations. For example, $AP4$, the generic assembly process for forming the family of end products from the immediately lower-level product items, involves two generic assembly operations AO_1^{AP4} and AO_2^{AP4} , while $MP1$, a generic manufacturing process, only has one generic manufacturing operation MO_1^{MP1} . The cycle time and setup of a process are the aggregation of these for all of its operations. In the case where each process only contains one operation, the process can be represented by the operation itself.

Figure 4.1 Concept implications of a process platform

4.2.1. Generic Variety Representation

A generic representation has been proven to be an effective means to describe a large number of variants with minimal data redundancy (Hegge and Wortmann, 1991; van Veen, 1992). An item is generic in the sense that it represents a set of similar items (namely variants) of the same type (namely a family). The item may be a component of the product (an end product, an assembly, an intermediate part, a purchased part, or raw material) or an operation of the routing (a manufacturing or assembly operation).

Instead of using part numbers (so called direct identification), the identification of individual variants from a generic item is based on variety parameters and their instances (a

list of parameter values). This is referred to as indirect identification (Hegge, 1992). Such an indirect identification entails a type of class-member relationship (exhibiting a meta-structure) between a family and its variants (Jiao et al., 2000). In this way, generic variety representation facilitates the specification of feasible variations of the items (products or routings) with respect to optional and alternative values of variety parameters.

4.2.2. Generic Routing Structure

Product data can be represented by a BOM that is used for an end product to state raw materials and intermediate parts or assemblies required for making the product. On the other hand, production information is concerned with how a product is produced, that is, the specification of operations sequences to be performed at corresponding work centers along with related resources such as machines, labors, tools, fixtures and setups. Similar to describing a product structure using a BOM, an operations routing can be constructed to represent the production structure for a given product (Jiao et al., 2000).

In general a product family of diverse product variants is characterized by a generic product structure (GPdS; Du et al., 2001), as shown in Figure 4.1(a). Accordingly, process data of all its product variants can be collated as a generic process structure (GPcS), as shown in Figure 4.1(b). The GPcS forms the basis of various routing variations in consequence of product variety.

The relationships between the product structure (i.e., BOMs) and the process structure are embodied in the materials required by particular operations. The link between BOM and process data can thus be established by specifying each component material in the BOM as required by the relevant process of the routing for making its parent product (Mather, 1987). Through these links, the GPdS and the GPcS can be synchronized into a unified generic structure, called generic routing structure (GRS) that is essential to and underpins the process platform. Therefore, as conceptually described in Figure 4.1(c), the GRS distinguishes the

common structure of the process platform, from which specific routings for given products are configured in the same way.

While the GPdS associates each component material directly with its parent product, a component material in the GRS is associated with the relevant process in the GPcS for producing its parent component. For each manufactured end or intermediate product, i.e., an item, a single-level GRS can be derived by specifying the sequence of the process required for producing that item in connection with the materials and resources required for the operations involved in the process. The multi-level GRS can be composed by linking the single-level GRSs of lower-level items through the processes that require them.

For example, in Figure 4.1, assume a variety parameter, *shape*, and its value set, $\{normal, special\}$, are associated with a generic component, *II*. The generic identification of *II* family is described as a set, $II \equiv \{II_1^*, II_2^*\}$. Thus two *II* variants can be identified using this variety parameter, i.e., $II_1^* = \{II | II.shape = "normal"\}$ and $II_2^* = \{II | II.shape = "special"\}$. The corresponding routing variation for producing *II* family involves two variants identified from a generic manufacturing process, *MPI*, which has a generic manufacturing operation, MO_i^{MPI} . To make II_1^* , a particular work center (W_{MPI*}), other than that (W_{MPI}) in the standard routing, has to be employed. In relation to W_{MPI*} the cycle time and the related setup along with the operation variant MO_i^{MPI*} become different for making II_1^* , i.e., the cycle time changing from 12.5 to 25.6 and the setup changing from S_{MPI} to S_{MPI*} , respectively.

4.2.3. Generic Planning for Production Configuration

Generic planning is introduced to determine specific process variations in routings in order to accommodate diverse product variants in the family. Within a process platform, the variations of routings result from the differences in product variants to be processed by operations in the routing. Therefore, configuration of routing variants from the GRS becomes

the major concern in generic planning. Taking advantage of the meta-structure inherent in the generic variety representation, routing configuration can be implemented through the instantiation of the GRS with respect to the given values of particular variety parameters, as shown in Figure 4.1(d).

Under the umbrella of the GRS, not only are the GPdS and the GPcS unified by the material requirement links, but also they employ exactly the same set of variety parameters and their values to handle variety (Jiao et al., 2000). Thus the class-member relationships between generic items and their variant sets can be consistently used for configuring routings for given product specifications. In this way, the correspondence between product and process variety can be maintained throughout the variation of both product and process structures. As shown in Figure 4.1, the same set of parameters, $\{C3.include, I2.include, I1.shape\}$, is used in generic planning for configuring routing variants in response to the specific product, that is,

Variety parameters and their values:

$\{C3.include=0, I2.include=0, I1.shape="special"\}$;

Product variant specification:

$\{I1 \xrightarrow{I1.shape="special"} I1^*, I2 \xrightarrow{I2.include=0} \emptyset, C3 \xrightarrow{C3.include=0} \emptyset\}$;

Routing variant configuration:

$\{MPI \xrightarrow{I1.shape="special"} MPI^*, MP2 \xrightarrow{I2.include=0} \emptyset, AP2 \xrightarrow{C3.include=0} AP2^*, AP4 \longrightarrow AP4\}$

During the process of configuring routings, every generic item involves an instantiation process, thus giving rise to a coordination issue among different types of instances of multiple generic items. Jiao et al. (2000) proposed a generic variety structure to coordinate multiple variants in regard to parameter values when exploding a GPdS. Accordingly, configuration constraints or planning rules are introduced to define relationships between parameter values of the product item variants and process elements, as well as relationships among process elements. The rules should guarantee for each valid specification of the

product item, precisely one valid specification of the related generic process and involved operations are generated through the configuration process.

4.3. Process Platform Formulation

In practice, production information for an enterprise is often described in various forms of policy documents such as product specifications, routing sheets, and job cards. These documents may be a suitable representation scheme for humans who must possess the knowledge to understand the information, but they do not lend themselves to formal analysis, monitoring, or improvement. Therefore, it is necessary to develop a modeling formalism. Such a formulation should provide a sufficiently powerful syntactic model to support rigorous analysis, simulation and manipulation of process platforms, while facilitating the application of semantics to support process platform enactment and detailed observations from a number of perspectives including customers, design and production (Mills, 1997).

Corresponding to the fundamentals of a process platform, process platform formulation involves three elements: the generic routing structure, generic variety representation and generic planning in production configuration as described next.

4.3.1. Generic Routing Structure

Definition 4-1: A *routing model* represents the operations of a routing as a set of operations, i.e., $\Lambda \equiv \{O_1, O_2, \dots, O_N\}$, where O_i , $i = 1, \dots, N$, denotes an individual *operation class*, may it be a type of manufacturing or assembly operation.

Definition 4-2: A GRS is defined as a tuple: $\Omega = \langle \Lambda, \succ \rangle$, where the precedence relationship $\succ$ describes the ordering of pairs of operations from the set Λ . A GRS, $\Omega \subset \Lambda \times \Lambda$, contains the pairs of operations from Λ that are related by $\succ$. While two operations with a precedence relationship are said to be sequential, two operations, one of

which does not rely on the completion of the other, are defined to be independent and thereby can be executed in parallel.

Definition 4-3: The *precedence relation* $\succ$ is defined such that a transitive closure of $\succ$ is irreflexive and $((O, X) \in \succ) \wedge ((O, Y) \in \succ) \Rightarrow X = Y$, where O, X and $Y \in \mathcal{A}$. This means with $\mathcal{A}$ the $\succ$ forms trees (called precedence graphs), and thus the GRS can be described as routings.

For illustrative simplicity, all processes in the original GRS in Figure 4.1(c) are assumed to contain one operation only, and thus the resulting GRS is given in Figure 4.2 together with its representation graph. The GRS in Figure 4.2(a) can be modeled as $\Omega = \langle \mathcal{A}, \succ \rangle$, where the set of operations $\mathcal{A}$ is defined as $\mathcal{A} = \{M1, M2, A1, A2, A3, A4\}$. With the addition of precedence relations, the GRS model Ω , represented in Figure 4.2(b) as a precedence graph, can thus be defined as: $\Omega = \{M1 \succ A1, A1 \succ A4, M2 \succ A4, A2 \succ A3, A3 \succ A4\}$.

Figure 4.2 Representation of a generic routing structure as a precedence graph

An operation with no predecessor is referred to as an initial operation, and the one with no successor is called a terminal operation. Operations that have one or more predecessor operations and at least one successor operation are referred to as intermediate operations. Within the context of a GRS, the routings may have more than one initial operation,

indicating possible parallel operations, but they possess only one terminal operation, corresponding to the final assembly. In addition, the case that an operation has more than one successor is excluded. This conforms to the convergence nature of assembly operations. In Figure 4.2(b), operations $M1$, $M2$ and $A2$ are initial operations, operations $A1$ and $A3$ are intermediate operations, and operation $A4$ is the terminal operation.

Two operations, O_x and O_y , such that $O_x \succ O_y$ are said to be sequential. For a GRS, $\Omega = \langle A, \succ \rangle$, a set of sequential operations, $A_s \in A$, and the associated precedence relations, $\succ_s \in \succ$, containing an initial operation, $O_\alpha \in A_s$, and a terminal operation, $O_\beta \in A_s$, is referred to as a chain. A GRS, $\Omega = \langle A, \succ \rangle$, may contain several chains. As shown in Figure 4.2(b), the precedence graph of GRS Ω contains three independent chains: $\Omega_x = \{M1 \succ A1, A1 \succ A4\}$, $\Omega^y = \{M2 \succ A4\}$, and $\Omega_z = \{A2 \succ A3, A3 \succ A4\}$.

In addition to the precedence issue, the characteristics of each operation need to be modeled, namely the operation description, as defined next.

Definition 4-4: The material flow of each operation is distinguished in terms of component classes. A *component class* is defined as a tuple: $C = \langle Co, Q \rangle$, where Co denotes the type of component part and Q indicates the number of units of this type of component part required for producing one unit of its immediate parent component class (i.e., the so-called *quantity per* of a goes-into relationship in a BOM structure). As illustrated in Figure 4.1, a component class may be of an end product, a subassembly, an intermediate part or raw material type.

Definition 4-5: Associated with each operation, $O \in A$, is an *operation description* such that, $\Psi_O = \langle \Phi_O^M, \Phi_O^P, R_O \rangle$, where Φ_O^M , Φ_O^P and R_O are called the material, product and resource classes of operation O , respectively.

Definition 4-6: A *material class* of an operation, $O \in \Lambda$, is a set of component classes, i.e., $\Phi_O^M = \{C_{O_i} \mid i=1, 2, \dots, r_O\}$, $\forall C_{O_i} \sim C$. It refers to the types of input materials that will be *transformed* to their parent class by the operation.

Definition 4-7: A *product class* of an operation, $O \in \Lambda$, is one of component classes, i.e., $\Phi_O^P \equiv \{C_{OP}\}$, $\exists C_{OP} \sim C$, $Q_{OP} = 1$. A product class denotes one type of product that will be *produced* by the operation. Usually an operation produces only one type of product, and may require multiple types of materials. In addition, the quantity per of the component class associated with a product class is always equal to one unit.

While Φ_O^M and Φ_O^P constitute the material flow through operation O , a resource class, R_O , characterizes the types of manufacturing resources required by the operation in order to produce Φ_O^P .

Definition 4-8: A *resource class* of an operation, $O \in \Lambda$, is defined as a triplet: $R_O = \langle W_O, T_O, S_O \rangle$, where W_O , T_O and S_O stand for the work center, cycle time and setup classes that are *consumed* by the operation, respectively. A *work center class* denotes the type of machines or work stations to be used for an operation. The *cycle time class* of an operation suggests variation of the cycle time of an operation under different production conditions. The rationale of relating cycle times to variety parameters within a process platform is discussed in Jiao and Tseng (2004). In practice, the cycle time is often estimated based on the principle of establishing standard times for work measurement and time study (Hodson, 1992). A *setup class* represents various types of tooling and/or fixture related to an operation. A setup class can be described by a tooling or fixture class, or both of them.

Figure 4.3 illustrates the class definitions and their interrelationships within a GRS. The definitions of component and operation classes in the context of process platform enable inheritance among component and operations classes (i.e., the generic components and operations) and facilitate hierarchical composition among the instances of component and

operation classes (i.e., the specific component and operation variants). The class-member relationships between generic components and operations and their variants are formulated as generic variety representation in the next section.

Figure 4.3 Classes and their interrelationships within a generic routing structure

4.3.2. Generic Variety Representation

The foundation of generic variety representation originates from object-oriented modeling. Along with behavior and identity, the concept of state is essential to the definition of an object in object-oriented modeling. Usually the state of an object encompasses all static properties of the object as well as the dynamic values of these properties. As shown in Figure 4.3, every object class defined in a GRS can be regarded as a generic item. If these generic items are assumed to be analogous to the objects in object-oriented modeling, a concept of variety state should also hold true in playing a similar role in indirect identification of individual variants from generic items.

In this regard, a *variety handler and its states* are introduced as an abstract way to represent and reason about the characteristics of individual variants as well as their differentiation with respect to each generic item. The assumption behind a variety handler and its states is that certain properties of each component class can be characterized as distinguishable attributes, i.e., so-called variety parameters (Jiao et al., 2000). These variety parameters are attached to various primitive components (i.e., those component parts

excluding intermediate parts as shown in Figure 4.1) (Du et al., 2001). In addition, variety of a compound module (i.e., an intermediate part as shown in Figure 4.1) results from propagation of the variety of its child components (Du et al., 2001). Accordingly, variety handlers are defined for those non-intermediate-part component classes only, and the variety handler of a product class comprises all variety handlers of its child component classes. While each variety handler associates a type of component (i.e., component class) with a set of variety parameters, the states of the handler relate specific variants to the values of particular variety parameters.

Definition 4-9: In a GRS as shown in Figure 4.1(c), assume all component classes form a *component class set*: $\Gamma \equiv \{C_1, C_2, \dots, C_M\}$, where $C_i | i = 1, \dots, M$ denotes a component class (refer to Definition 4-4). Further assume: $\Gamma^{IP} \equiv \{C_1^{IP}, C_2^{IP}, \dots, C_{r^{IP}}^{IP}\}$, where $C_i^{IP} | i = 1, \dots, r^{IP}$, $r^{IP} < M$, denotes a component class that is a type of intermediate part, may it be the product class of a manufacturing or an assembly operation (refer to Definition 4-7). Therefore, a *primitive component class set* can be defined as: $\Gamma^{Pri} = \Gamma \cap \overline{\Gamma^{IP}}$, in which a *primitive component class* denotes a component class that is not of the type of intermediate parts (i.e., $\Gamma^{Pri} \cap \Gamma^{IP} = \emptyset$).

Definition 4-10: For a primitive component class, $C_x \in \Gamma^{Pri} \subset \Gamma$, assume a set of variety parameters, $A_x \equiv \{\alpha_{x0}, \alpha_{x1}, \dots, \alpha_{xm}\}$, is identified to represent the variety characteristics of C_x . The *variety handler* of component class C_x is formally defined as the vector of variety parameter set A_x , i.e., $\Theta_x = \overrightarrow{A_x} \equiv [\alpha_{x0}, \alpha_{x1}, \dots, \alpha_{xm}]$.

For each variety parameter of C_x , $\alpha_{xk} \in A_x$, $\forall k \in \{0, 1, \dots, m\}$, usually there exists more than one alternative value, i.e., $A_{xk}^* \equiv \{\alpha_{xk1}^*, \dots, \alpha_{xkj}^*, \dots, \alpha_{xkn_k}^*\}$, where A_{xk}^* denotes the value set of variety parameter α_{xk} , and n_k is the number of values that can be assumed by α_{xk} .

As a notational convention, the superscript ‘*’ is used within generic variety representation to denote the variant of a generic item, the instance of an object class, or the value of a variety parameter. For a component variant, $y \in C_x^*$, where C_x^* is the set of component variants of component class (i.e., generic component) C_x , its variety characteristics are inherited from C_x . This means the variety handler Θ_x of component class of type x postulates the instantiation of variety parameters into individual component variants of the same type. Therefore, inside the component variant y , each variety parameter, corresponding to the variety handler of its component class, has an associated value, λ_{yk} , such that $\exists \lambda_{yk} \in A_{xk}^*$. In other words, the values that can be assumed by λ_{yk} forms a set such that,

$$A_{yk}^* \equiv \{\lambda_{yk1}, \dots, \lambda_{ykn_k}, \dots, \lambda_{ykn_k}\} \xleftarrow{\{\lambda_{yjk} = \alpha_{xjk}^* \mid j=1, \dots, n_k\}} \{\alpha_{xk1}^*, \dots, \alpha_{xkj}^*, \dots, \alpha_{xkn_k}^*\} \equiv A_{xk}^*. \quad (4.1)$$

Definition 4-11: Based on the above concepts and definitions, a *state of variety handler* $\Theta_x \equiv [\alpha_{x0}, \alpha_{x1}, \dots, \alpha_{xm}]$ in relation to a component variant, $y \in C_x^*$, can be formally defined as the vector of a specific value set, $A_y^* \equiv \{\lambda_{y0}, \lambda_{y1}, \dots, \lambda_{ym}\}$, that describes the variety characteristics of y , i.e., $\Theta_{xy}^* = \overrightarrow{A_y^*} \equiv [\lambda_{y0}, \lambda_{y1}, \dots, \lambda_{ym}]$, $\exists \lambda_{yk} \in A_{xk}^*$, $\forall k \in \{0, 1, \dots, m\}$. The variety handler state Θ_{xy}^* can thus be regarded as an element of the set of variety parameter values that can be assumed by an individual component variant, y , that is, $A_y^* \in \{A_{x0}^* \times A_{x1}^* \times \dots \times A_{xm}^*\}$.

Proposition 4-1: Given an operation, $O \sim \Psi_O = \langle \Phi_O^M, \Phi_O^P, R_O \rangle$, the variety handler of its product class, Θ^P , is a composite vector of all variety handlers of those component classes from its material class, Θ_g^M , $\forall g \in \{1, \dots, K\}$, that is, $\Theta^P = \bigcup_{g=1}^K \Theta_g^M = [\Theta_1^M, \Theta_2^M, \dots, \Theta_K^M]$, where K is the number of component classes constituting the material class Φ_O^M .

Proof 4-1: Assume the material class of, for example, an initial operation class consists of the following primitive component classes: $\Phi_O^M \equiv \{C_l, \dots, C_g, \dots, C_K\}$, $\forall C_g \in \Gamma^{Pri} \subset \Gamma$. Within Φ_O^M , each component class is associated with certain variety parameters that form a set, i.e.,

$$\begin{aligned} A_l^M &\equiv \{\alpha_{l0}^M, \alpha_{l1}^M, \dots, \alpha_{lm_l}^M\}, \\ A_g^M &\equiv \{\alpha_{g0}^M, \alpha_{g1}^M, \dots, \alpha_{gm_g}^M\}, \\ A_K^M &\equiv \{\alpha_{K0}^M, \alpha_{K1}^M, \dots, \alpha_{Km_K}^M\}. \end{aligned} \quad (4.2)$$

where m_g is the number of variety parameters associated with C_g . Since both product and material classes are sub-classes of the component class (refer to Figure 4.3) and all component classes constituting the material class suppose to be transformed to the product class through operation O , the product class inherits all the variety parameters of these component classes, which is referred to as variety propagation through nesting (Du et al., 2001). As a result, the variety parameter set of a product class is given as $A^P = A_l^M \cup \dots A_g^M \cup \dots A_K^M$. Accordingly, the variety handler of a product class is formally defined as a vector of its variety parameter set, that is,

$$\Theta^P = \overrightarrow{A^P} \equiv [\alpha_{l0}^M, \alpha_{l1}^M, \dots, \alpha_{lm_l}^M, \dots, \alpha_{g0}^M, \alpha_{g1}^M, \dots, \alpha_{gm_g}^M, \dots, \alpha_{K0}^M, \alpha_{K1}^M, \dots, \alpha_{Km_K}^M].$$

Proposition 4-2: Based on Definition 4-11, the state of variety handler in relation to a product variant, $z \in C_{OP}^*$, is a composite vector of the individual variety handler states of a specific set of component material variants associated with z , that is,

$$\Theta_z^{P*} = \bigcup_{g=l}^K \Theta_{zg}^{M*} = [\Theta_{zl}^{M*}, \Theta_{z2}^{M*}, \dots, \Theta_{zK}^{M*}], \text{ where } \Theta_{zg}^{M*}, \forall g \in \{l, \dots, K\}, \text{ denotes a specific set of the}$$

variety handler states of those component variants to be transformed to the product variant z through a particular operation (variant), O_z^* .

Proof 4-2: Associated with Θ_z^{P*} is a specific value set, $A_z^{P*} \equiv \{\lambda_{zI}, \dots, \lambda_{zS}\}$, that describes the variety characteristics of z , i.e., $\Theta_z^{P*} = \overrightarrow{A_z^{P*}} \equiv [\lambda_{zI}, \dots, \lambda_{zS}]$, $\exists \lambda_{zw} \in \{A_{zgj}^{M*} \mid g=1, \dots, K; j=0, 1, \dots, m_g\}$, $\forall w \in \{I, \dots, S\}$, $S = \sum_{g=1}^K (m_g + 1)$, where A_{zgj}^{M*} denotes a specific value set of a variety parameter, α_{zgj}^M , $\exists j \in \{0, 1, \dots, m_g\}$, of C_{zg}^* associated with specific product variant z . The variety handler state Θ_z^{P*} of product variant z thus consists of a series of elements from specific sets of variety parameter values that can be assumed by individual component material variants, $\{C_{zg}^* \mid g=1, \dots, K\}$, that is,

$$A_z^{P*} \in \{A_{zI0}^{M*} \times A_{zI1}^{M*} \times \dots \times A_{zIm_I}^{M*} \times \dots \times A_{zg0}^{M*} \times A_{zg1}^{M*} \times \dots \times A_{zgm_g}^{M*} \times \dots \times A_{zK0}^{M*} \times A_{zK1}^{M*} \times \dots \times A_{zKm_K}^{M*}\}. \quad (4.3)$$

Corollary 4-1: For an operation class, $O \in \mathcal{A}$, its variety handler, Θ^O , is a composite vector of all variety handlers of those component classes from the material class associated with the operation, Θ_g^M , $\forall g \in \{1, \dots, K\}$, that is, $\Theta^O = [\Theta_1^M, \Theta_2^M, \dots, \Theta_K^M]$, where K is the number of component classes constituting the material class Φ_O^M .

Corollary 4-2: The state of variety handler in relation to an operation variant, $\zeta \in O^*$, is a composite vector of the individual variety handler states of a specific set of component material variants associated with ζ , that is, $\Theta_\zeta^{O*} = [\Theta_{\zeta 1}^{M*}, \Theta_{\zeta 2}^{M*}, \dots, \Theta_{\zeta K}^{M*}]$, where $\Theta_{\zeta g}^{M*}$, $\forall g \in \{1, \dots, K\}$, denotes a specific set of the variety handler states of those component material variants to be transformed through particular operation (variant), ζ .

Corollary 4-3: Within a GRS, the variety characteristics of all objects associated with an operation can be modeled using a unified variety handler and its states of their origins – the operation itself. This is due to inheritance among object classes defined in a GRS (refer to Figure 4.3).

In general, given an operation class, $O \sim \Psi_O = \langle \Phi_O^M, \Phi_O^P, R_O \rangle$, $R_O = \langle W_O, T_O, S_O \rangle$, and one of its variants, $\zeta \in O^*$, Corollary 4-3 implies the following,

$$\begin{aligned} \Theta^W &= \Theta^T = \Theta^S = \Theta^R = \Theta^P \equiv \Theta^O, \text{ and} \\ \Theta_\zeta^{W*} &= \Theta_\zeta^{T*} = \Theta_\zeta^{S*} = \Theta_\zeta^{R*} = \Theta_\zeta^{P*} \equiv \Theta_\zeta^{O*}, \end{aligned} \quad (4.4)$$

where Θ^W , Θ^T , Θ^S , Θ^R , Θ^P , and Θ^M denote the respective variety handlers of the work center, cycle time, setup, resource, product, and material classes associated with the operation class O ; Θ_ζ^{W*} , Θ_ζ^{T*} , Θ_ζ^{S*} , Θ_ζ^{R*} , Θ_ζ^{P*} and $\Theta_{\zeta_g}^{M*}$ refer to the respective variety handler states of an individual specific work center, cycle time, setup, resource, product, and material variants related to the particular operation variant ζ ; and Θ^O and Θ_ζ^{O*} stand for the variety handler of the operation class O and its state related to the operation variant ζ , respectively. As a result, variety coordination within a process platform can be achieved through consistent use of the variety handler and its states for an operation throughout various objects related to the operation.

Furthermore, the variety handler and its states for an operation are defined in terms of respective composite vectors of all variety handlers and their states of those component materials associated with the operation. This means that the variety of component materials contributes to the source of various process variety related to an operation, conforming to the general observation that an operation is essentially meant to transform certain component materials to certain products. As a result, variety of those primitive components grounds various variety and their propagation within a process platform.

4.3.3. Generic Planning

A specific operation variant, $\zeta \sim O_\zeta^* \in O^*$, is distinguished by its operation description, $\Psi_\zeta^* = (\Phi_\zeta^{M*}, \Phi_\zeta^{P*}, R_\zeta^*)$. The materials to be processed by the operation are given by a

component material set, $\Phi_{\zeta}^{M*} = \{C_{\zeta y}^* \mid y=1, \dots, M_{\zeta}\}$, comprising a set of component variants, $C_{\zeta y}^* \in \Gamma$, $\forall y \in \{1, \dots, M_{\zeta}\}$. These component variants may present diversity in terms of component part $Co_{\zeta y}^*$ and/or the corresponding quantity per $Q_{\zeta y}^*$. Differentiations of each component variant in regard to $Co_{\zeta y}^*$ and $Q_{\zeta y}^*$ can be characterized by the specific state of its variety handler, $\Theta_{\zeta y}^{C*} \equiv [\lambda_{\zeta y 0}, \lambda_{\zeta y 1}, \dots, \lambda_{\zeta y m_y}]$. The product variant of the operation is an intermediate part transformed from these component material variants, i.e., $\Phi_{\zeta}^{P*} \equiv \{C_{\zeta P}^*\}$, $C_{\zeta P}^* \in \Gamma^{IP}$. The particular resource variant required by the operation can be elaborated in terms of the specific work center, cycle time, and setup variants, i.e., $R_{\zeta}^* \equiv (W_{\zeta}^*, T_{\zeta}^*, S_{\zeta}^*)$. All variety characteristics of $Co_{\zeta P}^*$, W_{ζ}^* , T_{ζ}^* and S_{ζ}^* can be detected by the variety handler state of the operation, Θ_{ζ}^* , which is a composite of individual variety handler states of the associated component material variants, i.e., $\Theta_{\zeta}^* = \bigcup_{y=1}^{M_{\zeta}} \Theta_{\zeta y}^{C*}$. Therefore, the variety handler states of a specific operation become the unique identifiers to various process variety, in terms of component materials and their quantities per, products, work centers, cycle times, and setups, in relation to the operation. The correspondence between operation variety handler states and operation-related variety can be organized as a process variety grid, as shown in Table 4.1.

Definition 4-12: Within the context of a process platform, each component possesses at least one variety characteristic – if it is included or not in an operation to produce a parent component. As such, variety parameter $\alpha_{x0} \in A_x$ (refer to Definition 4-10) is reserved for this purpose and thus is referred to as the *cardinal flag* of component class $C_x \in \Gamma$. Only two values, 1 or 0, are possible for α_{x0} , i.e., $A_{x0}^* = \{\alpha_{x0}^*\} = \{1, 0\}$, suggesting that:

Table 4.1 Process variety grid

ζ	Operation Variant $O_{\zeta}^* = \{C_{\zeta}^* \mid_{\forall y} \}, C_{\zeta}^*, R_{\zeta}^*\}$		Component variants $C_{\zeta}^* \equiv (Co_{\zeta}^*, Q_{\zeta}^*), \forall y \in \{1, \dots, M_{\zeta}\}$				Product variant $C_{\zeta}^* \equiv (Co_{\zeta}^*, Q_{\zeta}^*)$		Resource variant $R_{\zeta}^* \equiv (W_{\zeta}^*, T_{\zeta}^*, S_{\zeta}^*)$		
		y	State of variety handler $\Theta_{\zeta}^* \equiv [\lambda_{\zeta 0}, \lambda_{\zeta 1}, \dots, \lambda_{\zeta m_{\zeta}}]$	Component part Co_{ζ}^*	Quantity per Q_{ζ}^*	Component part Co_{ζ}^*	Quantity per Q_{ζ}^*	Component part Co_{ζ}^*	Work-center W_{ζ}^*	Cycle time T_{ζ}^*	Setup S_{ζ}^*
1	State of variety handler $\Theta_{\zeta}^* = [\Theta_{\zeta 1}^*, \Theta_{\zeta 2}^*, \dots, \Theta_{\zeta M_{\zeta}}^*]$	1	$[\lambda_{110}, \lambda_{111}, \dots, \lambda_{11m_1}]$	Co_{11}^*	Q_{11}^*	Co_{11}^*	Q_{11}^*	Co_{1P}^*	W_1^*	T_1^*	S_1^*
		2	$[\lambda_{120}, \lambda_{121}, \dots, \lambda_{12m_2}]$	Co_{12}^*	Q_{12}^*	Co_{12}^*	Q_{12}^*	Co_{1P}^*	W_1^*	T_1^*	S_1^*
		...	...	...	...	...	...	Co_{1P}^*	W_1^*	T_1^*	S_1^*
		M_1	$[\lambda_{1M_1 0}, \lambda_{1M_1 1}, \dots, \lambda_{1M_1 m_{M_1}}]$	$Co_{1M_1}^*$	$Q_{1M_1}^*$	$Co_{1M_1}^*$	$Q_{1M_1}^*$	Co_{1P}^*	W_1^*	T_1^*	S_1^*
2	State of variety handler $\Theta_{\zeta}^* = [\Theta_{\zeta 1}^*, \Theta_{\zeta 2}^*, \dots, \Theta_{\zeta M_{\zeta}}^*]$	1	$[\lambda_{210}, \lambda_{211}, \dots, \lambda_{21m_1}]$	Co_{21}^*	Q_{21}^*	Co_{21}^*	Q_{21}^*	Co_{2P}^*	W_2^*	T_2^*	S_2^*
		2	$[\lambda_{220}, \lambda_{221}, \dots, \lambda_{22m_2}]$	Co_{22}^*	Q_{22}^*	Co_{22}^*	Q_{22}^*	Co_{2P}^*	W_2^*	T_2^*	S_2^*
		...	...	...	...	...	...	Co_{2P}^*	W_2^*	T_2^*	S_2^*
		M_2	$[\lambda_{2M_2 0}, \lambda_{2M_2 1}, \dots, \lambda_{2M_2 m_{M_2}}]$	$Co_{2M_2}^*$	$Q_{2M_2}^*$	$Co_{2M_2}^*$	$Q_{2M_2}^*$	Co_{2P}^*	W_2^*	T_2^*	S_2^*
...	...	...	...	...	...	...	...	...	...	...	...
N	State of variety handler $\Theta_{\zeta}^* = [\Theta_{\zeta 1}^*, \Theta_{\zeta 2}^*, \dots, \Theta_{\zeta M_{\zeta}}^*]$	1	$[\lambda_{N10}, \lambda_{N11}, \dots, \lambda_{N1m_N}]$	Co_{N1}^*	Q_{N1}^*	Co_{N1}^*	Q_{N1}^*	Co_{NP}^*	W_N^*	T_N^*	S_N^*
		2	$[\lambda_{N20}, \lambda_{N21}, \dots, \lambda_{N2m_N}]$	Co_{N2}^*	Q_{N2}^*	Co_{N2}^*	Q_{N2}^*	Co_{NP}^*	W_N^*	T_N^*	S_N^*
		...	...	...	...	...	...	Co_{NP}^*	W_N^*	T_N^*	S_N^*
		M_N	$[\lambda_{NM_N 0}, \lambda_{NM_N 1}, \dots, \lambda_{NM_N m_{M_N}}]$	$Co_{NM_N}^*$	$Q_{NM_N}^*$	$Co_{NM_N}^*$	$Q_{NM_N}^*$	Co_{NP}^*	W_N^*	T_N^*	S_N^*

$$\left\{ \begin{array}{l} \alpha_{x0}^* = 1: \text{Component variant } C_x^* \text{ is to be} \\ \quad \text{included in a parent component through an operation} \\ \alpha_{x0}^* = 0: \text{Component variant } C_x^* \text{ is not to be} \\ \quad \text{included in a parent component} \end{array} \right.$$

Corollary 4-4: Given an intermediate part, $C_x^{IP} \in \Gamma^{IP}$, consisting of a set of child components, $C_{xg} \in \Gamma$, $g = 1, \dots, K$. The cardinal flag of C_x^{IP} is determined by the cardinal flags of all its child components: $\alpha_{x10} \wedge \dots \wedge \alpha_{xg0} \wedge \dots \wedge \alpha_{xK0}$. This implies that the intermediate part will not be included in its parent component (another intermediate part) if it contains no child component of itself.

Corollary 4-5: Given an operation, O_X , with a set of component materials, $\Phi_X^M = \{C_{Xg} \mid g=1, \dots, K\}$, $C_{Xg} \in \Gamma$. The cardinal flag of the operation is determined by the cardinal flags of all its component materials: $\alpha_{X10} \wedge \dots \wedge \alpha_{Xg0} \wedge \dots \wedge \alpha_{XK0}$. It thus indicates that an operation is one of the valid operations included in a specific operations routing depends on whether it is to be used to transform certain materials, i.e., if the set of component materials is empty, or not.

Definition 4-13: Given a generic operation, O_X , with class of X , the derivation of its variants is achieved through instantiation of the generic operation according to specific variety parameter values. The set of specific operation variants is defined as:

$$O_X^* = \{O_\zeta^* \mid O_\zeta^* \text{ is an operation possessing the characteristics of } X\}. \quad (4.5)$$

This definition renders the capability to handle the derivation of multiple variants from a generic operation, $O_X \in \Lambda$. Such a repeated derivation causes O_X to be replicated within the framework of an executing GRS. Each replicated operation, O_X' , is renamed and added to Λ , so as to preserve the anti-symmetric and anti-reflexive nature of the precedence relation $\succ$, as well as to maintain the acyclic nature of the corresponding precedence graph $\Omega = \langle \Lambda, \succ \rangle$; but is otherwise identical to O_X .

Definition 4-14: In generic planning, variant configuration is incarnated through the assignment of different types of variants related to a specific operation variant, $O_\zeta^* \in O_X^*$, to the respective *domains* for operation O_X . Each operation domain contains a list of specific variants derived from O_X . Corresponding to the operation description $\Psi_X = (\Phi_X^M, \Phi_X^P, R_X)$, three domains are introduced to operation O_X :

(1) *Material domain*, $D_X^M = \{(\Theta_{\zeta y}^{C^*}, Co_{\zeta y}^*, Q_{\zeta y}^*) | y = 1, \dots, M_\zeta\}$, denotes a list of component material variants assumed by O_ζ^* . Each component material variant is described by a triplet of the associated variety handler state, the component part variant and its specific quantity per, i.e., $C_{\zeta y}^* \sim (\Theta_{\zeta y}^{C^*}, Co_{\zeta y}^*, Q_{\zeta y}^*)$, $\forall y \in \{1, \dots, M_\zeta\}$;

(2) *Product domain*, $D_X^P \equiv \{(\Theta_\zeta^*, Co_{\zeta P}^*, I)\}$, is a set consisting of the product variant of O_ζ^* described by a triplet of the associated variety handler state, the specific component part variant, and its quantity per, i.e., $C_{\zeta P}^* \sim (\Theta_\zeta^*, Co_{\zeta P}^*, I)$;

(3) *Resource domain*, $D_X^R \equiv \{(\Theta_\zeta^*, W_\zeta^*, T_\zeta^*, S_\zeta^*)\}$, represents the resource variant of O_ζ^* described by a quadlet of the associated variety handler state, the particular work center, cycle time, and setup variants, i.e., $R_\zeta^* \sim (\Theta_\zeta^*, W_\zeta^*, T_\zeta^*, S_\zeta^*) \in R_X^*$.

Corollary 4-6: If an operation O_X is not to be included in a specific routing, then it means that there is no specific variant of this operation. This can be indicated by the empty domains of O_X , i.e., $D_X^M = D_X^P = D_X^R = \emptyset$.

Corollary 4-7: A precedence relation, $O_X \succ O_{X'}$, holds true if the product of O_X becomes one of the component materials of $O_{X'}$, that is, $O_X \succ O_{X'} : \text{iff } D_{X'}^M \neq \emptyset \text{ and } D_X^P \neq \emptyset$;
 $\exists C_{\zeta' y}^* \in D_{X'}^M$ such that $C_{\zeta' y}^* \cdot \Theta_{\zeta' y}^{C^*} \equiv C_{\zeta P}^* \cdot \Theta_\zeta^*$, where $C_{\zeta' y}^* \sim (\Theta_{\zeta' y}^{C^*}, Co_{\zeta' y}^*, Q_{\zeta' y}^*)$, $\forall y \in \{1, \dots, M_{\zeta'}\}$,
 $C_{\zeta P}^* \in D_X^P$, $C_{\zeta P}^* \sim (\Theta_\zeta^*, Co_{\zeta P}^*, I)$.

In the context of mass customization, each customized product is characterized by specific values of a set of variety parameters (Jiao et al., 2000). A process platform supports the fulfillment of customized products through the configuration of corresponding routings. Generic planning is such a mechanism of mapping the individual products to its routing, $\Omega^* = \langle A^*, \succ \rangle$, within a GRS, from which specific operations are determined. The procedure of generic planning in production configuration consists of the following steps:

(1) *Initialization.* Given a customized product, it is described as a set of variety parameters and their values: $\{(\beta_i, \beta_i^*) | i=1, \dots, L\}$. For a GRS, $\Omega = \langle A, \succ \rangle$, with an established generic variety structure, a special symbol ‘†’, meaning “not included”, is assigned as the default value to the state of every variety handler ($\{\Theta_\zeta | \zeta=1, \dots, N\}$ and $\{\Theta_{\zeta y}^C | \zeta=1, \dots, N; y=1, \dots, M_\zeta\}$). For each $O_X \in A$, let $D_X^M = D_X^P = D_X^R = \emptyset$. Further let $\Omega^* = \langle A^*, \succ \rangle = \emptyset$.

(2) *Variety Handler State Specification.* Based on planning or configuration rules, construct a process variety grid that lists all feasible variants of each generic item of the GRS. Each variant (e.g., $Co_{\zeta y}^*$, $Q_{\zeta y}^*$, $Co_{\zeta p}^*$, W_ζ^* , T_ζ^* , S_ζ^*) is characterized by its variety handler state. A valid variety handler state is determined by a feasible combination of relevant variety parameter values. Usually the feasible combinations of variety parameter values are specified by defining, for example, include conditions (Du et al., 2001), planning rules (Jiao et al., 2000), or configuration constraints (Sanderson and Uzumeri, 1997).

Among many variants of those generic items of GRS, a few variants that are relevant to the given product are referred to as valid variants identified by their variety handler states. Through specification of variety handler states, only those variety parameters relevant to $\{\beta_i |_{\forall i}\}$ obtain valid values corresponding to $\{\beta_i^* |_{\forall i}\}$. The variety handler state specification algorithm proceeds as follows.

For each variety parameter α_{xj} of a variety handler, $\Theta_x \equiv [\alpha_{x0}, \alpha_{x1}, \dots, \alpha_{xm}]$
 For each variety parameter β_i of the given product $\{(\beta_i, \beta_i^*) |_{i=1, \dots, L}\}$
 If $\alpha_{xj} \equiv \beta_i$
 Set $\alpha_{xj}^* = \beta_i^*$ /* The value of α_{xj} changes from ' $\dagger$ ' to β_i^* */
 End If
 End For ($i = 1, \dots, L$)
 End For ($j = 0, 1, \dots, m$)

(3) *Instantiation of Generic Items.* Instantiation of generic items is achieved by specifying the states of the relevant variety handlers in accordance with the given product. Valid variants are identified from the process variety grid by matching their variety handler states with those valid variety handler states. A valid handler state, $\Theta_x^* \equiv [\alpha_{x0}^*, \alpha_{x1}^*, \dots, \alpha_{xm}^*]$, is indicated by $\alpha_{x0}^* \vee \alpha_{x1}^* \vee \dots \vee \alpha_{xm}^* \neq \dagger$ and $\exists \alpha_{xj}^* \in \{\beta_i^* |_{i=1, \dots, L}\}, \forall j \in \{0, 1, \dots, m\}$.

Subsequently, all valid variants of $C_{\zeta'}^*$, $C_{\zeta^P}^*$, and R_{ζ}^* are assigned to the respective domains of O_X . This causes the domains of those valid operations, $O_X^* \in \Lambda^*$, to be populated and thus to become non-empty; otherwise, operation domains remain empty.

(4) *Configuration of Routing Variant.* A routing variant, $\Omega^* = \langle \Lambda^*, \succ \rangle$, is constructed by introducing precedence relations to those valid operation variants. An algorithm for deriving a routing variant can be developed based on Corollary 4-7. A detailed description of the algorithm is as follows:

For each operation $O_X \in \Lambda$
 For each operation $O_{X'}$ such that $O_{X'} \in \Lambda$ and $O_X \neq O_{X'}$
 If $D_X^M \cup D_X^P \cup D_{X'}^M \cup D_{X'}^P \neq \emptyset$
 For each component $(\Theta_{X'y}^{C^*}, Co_{X'y}^*, Q_{X'y}^*) \in D_{X'}^M$
 If $\Theta_{X'y}^{C^*} \equiv \Theta_X^*, (\Theta_X^*, Co_{X^P}^*, l) \in D_X^P$
 Add $O_X \succ O_{X'}$ to Ω^*
 Add O_X and $O_{X'}$ to Λ^*
 End If
 End For ($y = 1, \dots, M_{X'}$)
 End If
 End For
 End For

4.4. Coding-Based Implementation of Variety Coordination

In a high-variety production environment, the kernel of effective product and production data management lies in the achievement of variety coordination – to effectively streamline customer-perceived variety all the way down to product and process variety. This research proposes to implement variety coordination within a process platform based on coding systems. The idea is based on Group Technology principles, where similar parts are classified into families and are organized by coding (Mitrofanov, 1966).

With the formulation of a process platform, a structured coding system can be developed using a hybrid of established coding and classification methods. The coding presents a format for organizing a process platform into catalogs of product data and production information, where product and process variety can be accurately recorded and traced. The identification of individual items in the coding system coincides with unique identifications of variety handlers through specification of their states. Figure 4.4 illustrates the implementation of variety handlers and their states within a coding system. While each variety parameter of a variety handler requires a unique code in the coding scheme, the number of digits of the variety parameter code depends on its size – the number of variety parameter instances. For example, a decimal or alphabetic coding scheme can be used for a variety parameter possesses less than 10 (e.g., α_{x0} and α_{xm}) or between 10 and 26 (e.g., α_{x1}) instances. For a variety parameter with a large number of instances, more than one digit (e.g., α_{x2}), or a hexadecimal number system, may be used in the coding scheme.

Figure 4.4 Coding-based implementation of variety handlers and their states

As shown in Figure 4.5, variety propagation within a process platform is achieved through nesting of variety handlers. Accordingly, the composition of coding follows the same principle.

Figure 4.5 Variety propagation through nesting of variety handlers and of coding

The coding of a variety handler is the compound coding of all the coding of its child variety handlers, which in turn becomes one of the constituents of the coding of its parent variety handler. Therefore, the coding of primitive components (including raw materials) comprises the basic constructs of the coding of various items within a process platform, and the coding for the end product as well as the related final operation represents the most comprehensive coding of the platform. As a result, the coding of variety handlers and their

nesting can clearly describe the sources of variety and the differentiation of generic items with minimal data redundancy.

4.5. Case Study

A case study is conducted in a company in Singapore producing mass customized vibration motors for mobile phones to evaluate the proposals in this dissertation. In order to meet diverse customer needs related to mobile phones, the design and production of vibration motors are typically custom built, resulting in an exponentially increased amount of product and process variety. The production environment of the company is a combination of assemble-to-order and make-to-order, wherein a number of common standard components have been stored in stock, and the production of vibration motors does not start until customer orders are received. While the external business environment is characterized by rapidly changing and unpredictable customer demands, the company's internal manufacturing environment is facing the challenge of diverse custom designs and enormous variations in routing planning.

4.5.1. Generic Routing Structure

Based on the analysis of historical data on the company's product fulfillment and existing manufacturing capabilities, the vibration motor product platform is constructed, and accordingly the associated standard routings are identified. Figure 4.6 shows the GRS for vibration motors. The operations necessary to produce a vibration motor are indicated in Table 4.2.

Figure 4.6 The generic routing structure for vibration motors

Table 4.2 Operation classes of in the generic routing structure of the vibration motors

Operation Name	Description
<i>Avm</i>	Assembly of vibration motors
<i>Amb</i>	Assembly of mainbodies
<i>Aaa</i>	Assembly of armatures
<i>Afa</i>	Assembly of frames
<i>Aba</i>	Assembly of brackets
<i>Aca</i>	Assembly of coils
<i>Mc</i>	Manufacturing of coils
<i>Mf</i>	Manufacturing of frames
<i>Mba</i>	Manufacturing of brackets A
<i>Mbb</i>	Manufacturing of brackets B
<i>Mt</i>	Manufacturing of terminals

For the vibration motor GRS, the set of operations is defined as:

$$\Lambda = \{Avm, Amb, Aaa, Afa, Aba, Aca, Mc, Mf, Mba, Mbb, Mt\}. \quad (4.6)$$

Taking into account the precedence relations, the GRS model for vibration motors can be described as:

$$\Omega = \{Mc \succ Aca, Aca \succ Aaa, Aaa \succ Amb, Mf \succ Afa, Afa \succ Amb, Mba \succ Aba, Mbb \succ Aba, Mt \succ Aba, Aba \succ Amb, Amb \succ Avm\}. \quad (4.7)$$

Among the operations, Mc, Mf, Mba, Mbb and Mt are initial operations, Aca, Aaa, Afa, Aba and Amb are intermediate operations, and Avm the terminal operation. Between the initial and terminal operations, there are five chains of operations that are implied by the Ω , as shown below,

$$\begin{aligned}\Omega_1 &= \{Mc \succ Aca, Aca \succ Aaa, Aaa \succ Amb, Amb \succ Avm\}, \\ \Omega_2 &= \{Mf \succ Afa, Afa \succ Amb, Amb \succ Avm\}, \\ \Omega_3 &= \{Mba \succ Aba, Aba \succ Amb, Amb \succ Avm\}, \\ \Omega_4 &= \{Mbb \succ Aba, Aba \succ Amb, Amb \succ Avm\}, \text{ and} \\ \Omega_5 &= \{Mt \succ Aba, Aba \succ Amb, Amb \succ Avm\}.\end{aligned}\tag{4.8}$$

Descriptions for the operations of $\mathcal{A}$ include the material class Φ_O^M , the product class Φ_O^P , and the resource class R_O for each operation $O \in \mathcal{A}$. Table 4.3 gives the description associated with each operation in the Ω of vibration motors.

4.5.2. Generic Variety Representation

Within a process platform, product differentiation and routing variations are coordinated using generic variety representation. Table 4.4 shows the variety parameters and their instances defined for those primitive components in the vibration motor GRS. The variety parameters of an intermediate part, such as an assembly, are supposed to be propagated from the variety parameters of its child components. For a type of raw material for a manufacturing operation (e.g., RMc of Mc), its variety parameters (e.g., MC.F, MC.L, and MC.W) are defined as the same as those of its immediate product (e.g., the Coil). As a result, differentiation of raw materials is modeled in the same way as that of components for an assembly operation, in which the material requirement link can always be embodied in various components and related quantities per.

Table 4.3 The operation descriptions associated with each operation class in the generic routing structure of the vibration motors

Operation Class (O)	Operation Description: $\Psi_o = \langle \Phi_o^u, \Phi_o^p, R_o \rangle$				
	Material Class: $\Phi_o^M = \{(C_{o_i}, Q_i) \mid i=1, \dots, n_o\}$	Product Class: $\Phi_o^P = \{(C_{op}, I)\}$	Resource Class: $R_o = (W_o, T_o, S_o)$		
	Component (C_{o_i})	Quantity per (Q_i)	Component (C_{op})	Work center (W_o)	Cycle Time (T_o) Setup (S_o)
Avm	Weight (W) Mainbody (MB) Rubber Holder (RH)	Qw Qmb Qrh	Vibration Motor (VM)	Caulking Machine (W-Avm)	T-Avm Sifting Jig & Caulking Head (S-Avm)
Amb	Armature Assembly (AA) Frame Assembly (FA) Bracket Assembly (AB)	Qaa Qfa Qab	Mainbody (MB)	Caulking Machine & Inserting Machine (W-Amb)	T-Amb Caulking Blade & Bracket Holder (S-Amb)
Aba	Bracket A (BA) Bracket B (BB) Terminal (T)	Qba Qbb Qt	Bracket Assembly (AB)	Fusing Machine (W-Aba)	T-Aba Inserter & Pressing Jig (S-Aba)
Afa	Frame (F) Magnet (M)	Qf Qm	Frame Assembly (FA)	Pressing Machine (W-Afa)	T-Afa FA Holders (S-Afa)
Aaa	Shaft (S) Coil Assembly (CA)	Qs Qca	Armature Assembly (AA)	Inserting Machine & Soldering Machine (W-Aaa)	T-Aaa Supporting Holder & Pallet (S-Aaa)
Aca	Tape (TP) Commutator (CM) Coil (C)	Qtp Qcm Qc	Coil Assembly (CA)	Workbench (W-Aca)	T-Aca Guiding Jig (S-Aca)
Mc	RMc (MC)	Qmc	Coil (C)	Winding Machine (W-Mc)	T-Mc Tray (S-Mc)
Mf	RMf (MF)	Qmf	Frame (F)	Stamping Machine (W-Mf)	T-Mf MF Die & MF Holder (S-Mf)
Mba	RMba (MBA)	Qmba	Bracket A (BA)	Injection Machine (W-Mba)	T-Mba Adjustor & Locator (S-Mba)
Mbb	RMbb (MBB)	Qmbb	Bracket B (BB)	Injection Machine (W-Mbb)	T-Mbb Adjustor & Prealigning Jig (S-Mbb)
Mt	RMt (MT)	Qmt	Terminal (T)	Cutting Machine (W-Mt)	T-Mt MT Die & MT Holder (S-Mt)

Table 4.4 Variety parameters and their instances defined for vibration motors

Immediate Product	Primitive Component	Variety Parameters	Variety Parameter Instances
Coil (C)	RMc (MC)	Include Flag (MC.F) Length (MC.L) Winding Mode (MC.W)	1 4mm, 6mm ABA, BFT, SST, ZMF
Frame (F)	RMf (MF)	Include Flag (MF.F) Outer Diameter (MF.D) Length (MF.L)	1 4mm, 5mm 8m, 10mm, 11mm
Bracket A (BA)	RMba (MBA)	Include Flag (MBA.F) Shape (MBA.S) Thickness (MBA.T)	1 T, U, O BAA, BAB
Bracket B (BB)	RMbb (MBB)	Include Flag (MBB.F) Shape (MBB.S) Width (MBB.W)	1 T, U, O 4.5mm, 5.5mm
Terminal (T)	RMt (MT)	Include Flag (MT.F) Pitch (MT.P)	1 TS, TL
Coil Assembly (CA)	Tape (TP) Commutator (CM)	Include Flag (TP.F) Include Flag (CM.F) Thickness (CM.T)	1, 0 1 CMA, CMB, CMC
Armature Assembly (AA)	Shaft (S)	Include Flag (S.F) Length (S.L) Material (S.M)	1 10mm, 12mm ALM, PVC
Frame Assembly (FA)	Magnet (M)	Include Flag (M.F) Outer Diameter (M.D)	1 3.5mm, 4.7mm
Vibration Motor (VM)	Weight (W) Rubber Holder (RH)	Include Flag (W.F) Radius (W.R) Wight (W.W) Include Flag (RH.F) Size (RH.S)	1 2.0mm, 2.5mm, 3.0mm, 3.2mm WION, WPVC 1, 0 HSS, HSL

As shown in Table 4.4, every component must contain a variety parameter indicating the Include Flag. Except those of components TP and RH, all Include Flags assume only one instance – 1, suggesting that these components as well as their associated parent components are mandatory for each vibration motor. On the other hand, Components TP and RH, with their Include Flags bearing a value of 1 or 0, are optional components.

In the context of generic variety representation, the meta-structure embodied by variety parameters and their instances are modeled by introducing variety handlers. For example, as shown in Table 4.4, the variety parameter set of RMc is given as $A_{MC} = \{MC.F, MC.L, MC.W\}$. One of the feasible combinations of the values of these variety parameters is $A_{MC_y}^* = \{1, 4, ZMF\}$. The variety handler of RMc can thus be defined as a vector of this variety parameter set, i.e., $\Theta_{MC} = \overrightarrow{A_{MC}} = [MC.F, MC.L, MC.W]$. Corresponding to the feasible variety parameter value set $A_{MC_y}^*$, a state of the variety handler can be defined as $\Theta_{MC_y}^* = [1, 4, ZMF]$. According to the propositions and corollaries defined in Section 4.3.3, the variety handlers and their states of product variety (involving Co_{mc} , Q_{mc} , and its immediate product C) and process variety (involving

manufacturing operation Mc , work center $W-Mc$, cycle time $T-Mc$, and setup $S-Mc$) associated with raw material RMc can be determined in a unified way – in terms of variety of primitive components, i.e.,

$$\Theta_{CoMc} = \Theta_{QMc} = \Theta_C = \Theta_{Mc} = \Theta_{W-Mc} = \Theta_{T-Mc} = \Theta_{S-Mc} = [\Theta_{MC}] = [MC.F, MC.L, MC.W], \text{ and}$$

$$\Theta_{CoMc}^* = \Theta_{QMc}^* = \Theta_{C_y}^* = \Theta_{Mc_y}^* = \Theta_{W-Mc_y}^* = \Theta_{T-Mc_y}^* = \Theta_{S-Mc_y}^* = [\Theta_{MC_y}^*] = [1, 4, ZMF].$$

Variety propagation is one of the effective mechanisms to manage variety with minimal data redundancy. Based on the proposed formulation of process platform, variety propagation is achieved through the nesting of variety handlers. For example, the variety handler of assembly operation Aca or its product CA is a composite of variety handlers of its material components: TP , CM and C . The variety handler of C is another composite of the variety handler of its material component RMc . That is,

$$\Theta_{Aca} = \Theta_{CA} = [\Theta_{TP}, \Theta_{CM}, \Theta_C] = [TP.F, CM.F, CM.T, MC.F, MC.L, MC.W] \quad , \quad \text{where} \quad \Theta_{TP} = \Theta_{Qp} = [TP.F] \quad ,$$

$$\Theta_{CM} = \Theta_{Qcm} = [CM.F, CM.T], \text{ and } \Theta_C = [\Theta_{MC}] = [MC.F, MC.L, MC.W]. \text{ Similarly, the state of variety}$$

handler Θ_{Aca} or Θ_{CA} becomes a composite of feasible states of the variety handlers of its constituent components. That is, $\Theta_{Aca_y}^* = \Theta_{CA_y}^* = [\Theta_{TP_y}^*, \Theta_{CM_y}^*, \Theta_{C_y}^*]$. Recursively using the nesting of variety handlers, variety handlers for all product and process variety within the vibration motor process platform, as shown in Table 4.5, can be determined.

Based on the nesting of variety handler states, various instances of process variety can be determined, as shown in Table 4.6. Throughout nesting, every variety handler state is in relation with a viable combination of certain variety parameter values. In relation to each variety handler state, a GT code can be assigned to differentiate the representation of various variants, as shown in Table 4.6. Coding-based implementation of variety handlers and their states excels in dealing with diverse types of variety parameter values, either numerical (e.g., 3.2mm), binary (e.g., 1/0) or symbolic (e.g., WPVC), within one unified metric.

Table 4.5 Variety handlers associated with various generic items within the generic routing structure of the vibration motors

Operation Class: $O \sim \langle \phi_O^M, \phi_O^P, R_O \rangle$		Material Class: $\phi_O^M = \{ \langle C_O, Q \rangle \}_{i=1,2,\dots,r_O}$		Product Class: $\phi_O^P = \{ \langle C_O, Q_O \rangle \}$		Resource Class: $R_O = (W_O, T_O, S_O)$	
Operation	Variety Handler ($\phi_O = \phi_P$)	Component	Variety Handler ($\phi_C = \phi_{C_i}, \forall i = 1, \dots, r_O$)	Component	Variety Handler ($\phi_P = [\phi_{P_i}, \phi_{P_j}, \dots, \phi_{P_r}]$)	Resource	Variety Handler ($\phi_P = \phi_P = \phi_P = \phi_P = \phi_P$)
Mt	$\phi_{Mt} \equiv \phi_T$	MT	$\phi_{Mt} = \phi_{Q_{Mt}} = [MT.F, MT.P]$	T	$\phi_T = [\phi_{Tt}] = [MT.F, MT.P]$	(W-Mt, T-Mt, S-Mt)	$\phi_{Tt} = \phi_P = \phi_P = \phi_P = \phi_P$
Mbb	$\phi_{Mbb} \equiv \phi_{BB}$	MBB	$\phi_{Mbb} = \phi_{Q_{Mbb}} = [MBB.F, MBB.S, MBB.W]$	BB	$\phi_{BB} = [\phi_{BB}] = [MBB.F, MBB.S, MBB.W]$	(W-Mbb, T-Mbb, S-Mbb)	$\phi_{Mbb} = \phi_P = \phi_P = \phi_P = \phi_P$
Mba	$\phi_{Mba} \equiv \phi_{BA}$	MBA	$\phi_{Mba} = \phi_{Q_{Mba}} = [MBA.F, MBA.S, MBA.T]$	BA	$\phi_{BA} = [\phi_{BA}] = [MBA.F, MBA.S, MBA.T]$	(W-Mba, T-Mba, S-Mba)	$\phi_{Mba} = \phi_P = \phi_P = \phi_P = \phi_P$
Mf	$\phi_{Mf} \equiv \phi_F$	MF	$\phi_{Mf} = \phi_{Q_{Mf}} = [MF.F, MF.D, MF.L]$	F	$\phi_F = [\phi_{Mf}] = [MF.F, MF.D, MF.L]$	(W-Mf, T-Mf, S-Mf)	$\phi_{Mf} = \phi_P = \phi_P = \phi_P = \phi_P$
Mc	$\phi_{Mc} \equiv \phi_C$	MC	$\phi_{Mc} = \phi_{Q_{Mc}} = [MC.F, MC.L, MC.W]$	C	$\phi_C = [\phi_{Mc}] = [MC.F, MC.L, MC.W]$	(W-Mc, T-Mc, S-Mc)	$\phi_{Mc} = \phi_P = \phi_P = \phi_P = \phi_P$
Aca	$\phi_{Aca} \equiv \phi_{CA}$	TP CM C	$\phi_{Tp} = \phi_{Q_{Tp}} = [TP.F]$ $\phi_{Cm} = \phi_{Q_{Cm}} = [CM.F, CM.T]$ $\phi_C = \phi_{Q_C} = [MC.F, MC.L, MC.W]$	CA	$\phi_{Ca} = [\phi_{Ca}, \phi_{Ca}, \phi_{Ca}] = [TP.F, CM.F, CM.T, MC.F, MC.L, MC.W]$	(W-Aca, T-Aca, S-Aca)	$\phi_{Aca} = \phi_P = \phi_P = \phi_P = \phi_P$
Aaa	$\phi_{Aaa} \equiv \phi_{AA}$	S CA	$\phi_S = \phi_{Q_S} = [S.F, S.L, S.M]$ $\phi_{Ca} = \phi_{Q_{Ca}} = [TP.F, CM.F, CM.T, MC.F, MC.L, MC.W]$	AA	$\phi_{Aa} = [\phi_{Aa}, \phi_{Aa}] = [S.F, S.L, S.M, TP.F, CM.F, CM.T, MC.F, MC.L, MC.W]$	(W-Aaa, T-Aaa, S-Aaa)	$\phi_{Aaa} = \phi_P = \phi_P = \phi_P = \phi_P$
Afa	$\phi_{Afa} \equiv \phi_{FA}$	F M	$\phi_F = \phi_{Q_F} = [MF.F, MF.D, MF.L]$ $\phi_M = \phi_{Q_M} = [M.F, M.D]$	FA	$\phi_{Fa} = [\phi_{Fa}, \phi_{Fa}] = [MF.F, MF.D, MF.L, M.F, M.D]$	(W-Afa, T-Afa, S-Afa)	$\phi_{Afa} = \phi_P = \phi_P = \phi_P = \phi_P$
Aba	$\phi_{Aba} \equiv \phi_{AB}$	BA BB T	$\phi_{Ba} = \phi_{Q_{Ba}} = [MBA.F, MBA.S, MBA.T]$ $\phi_{Bb} = \phi_{Q_{Bb}} = [MBB.F, MBB.S, MBB.W]$ $\phi_T = \phi_{Q_T} = [MT.F, MT.P]$	AB	$\phi_{Ab} = [\phi_{Ab}, \phi_{Ab}, \phi_{Ab}] = [MBA.F, MBA.S, MBA.T, MBB.F, MBB.S, MBB.W, MT.F, MT.P]$	(W-Aba, T-Aba, S-Aba)	$\phi_{Aba} = \phi_P = \phi_P = \phi_P = \phi_P$
Amb	$\phi_{Amb} \equiv \phi_{AB}$	AA FA AB	$\phi_{Aa} = \phi_{Q_{Aa}} = [S.F, S.L, S.M, TP.F, CM.F, CM.T, MC.F, MC.L, MC.W]$ $\phi_{Fa} = \phi_{Q_{Fa}} = [MF.F, MF.D, MF.L, M.F, M.D]$ $\phi_{Ab} = \phi_{Q_{Ab}} = [MBA.F, MBA.S, MBA.T, MBB.F, MBB.S, MBB.W, MT.F, MT.P]$	MB	$\phi_{Mb} = [\phi_{Mb}, \phi_{Mb}, \phi_{Mb}] = [S.F, S.L, S.M, TP.F, CM.F, CM.T, MC.F, MC.L, MC.W, MF.F, MF.D, MF.L, M.F, M.D, MBA.F, MBA.S, MBA.T, MBB.F, MBB.S, MBB.W, MT.F, MT.P]$	(W-Amb, T-Amb, S-Amb)	$\phi_{Amb} = \phi_P = \phi_P = \phi_P = \phi_P$
Avm	$\phi_{Avm} \equiv \phi_{VM}$	W MB RH	$\phi_W = \phi_{Q_W} = [W.F, W.W, W.R]$ $\phi_{Mb} = \phi_{Q_{Mb}} = [S.F, S.L, S.M, TP.F, CM.F, CM.T, MC.F, MC.L, MC.W, MF.F, MF.D, MF.L, M.F, M.D, MBA.F, MBA.S, MBA.T, MBB.F, MBB.S, MBB.W, MT.F, MT.P]$ $\phi_{Rh} = \phi_{Q_{Rh}} = [RH.F, RH.S]$	VM	$\phi_{Vm} = [\phi_{Vm}, \phi_{Vm}, \phi_{Vm}] = [W.F, W.W, W.R, S.F, S.L, S.M, TP.F, CM.F, CM.T, MC.F, MC.L, MC.W, MF.F, MF.D, MF.L, M.F, M.D, MBA.F, MBA.S, MBA.T, MBB.F, MBB.S, MBB.W, MT.F, MT.P, RH.F, RH.S]$	(W-Avm, T-Avm, S-Avm)	$\phi_{Avm} = \phi_P = \phi_P = \phi_P = \phi_P$

Table 4.6 Process variety grid for the coil assembly (Aca)

Component Variant: $C_{\zeta^*}^* = \{(Co_{\zeta^*}^*, Q_{\zeta^*}^*) _{\forall y \in \{I, \dots, M_{\zeta^*}\}}\}$					Operation Variant: $O_{\zeta^*}^* = (\{C_{\zeta^*}^* _{\forall y}\}, C_{\zeta^*}^*, R_{\zeta^*}^*)$						
ID (y)	Variety Handler		Component Part ($Co_{\zeta^*}^*$)	Quantity Per ($Q_{\zeta^*}^*$)	ID (ζ)	Variety Handler		Product Variant ($C_{\zeta^*}^*$)	Resource Variant: $R_{\zeta^*}^*$		
	State ($\theta_{\zeta^*}^*$)	GT Code				State ($\theta_{\zeta^*}^*$)	GT Code		Work center ($W_{\zeta^*}^*$)	Cycle Time ($T_{\zeta^*}^*$)	Setup ($S_{\zeta^*}^*$)
MC1	[I,4,ABA]	111	MC1	1	Mc1	[I,4,ABA]	111	C1	WMc	4.52	SMcA
MC2	[I,4,BFT]	112	MC2	1	Mc2	[I,4,BFT]	112	C2	WMc	4.78	SMcB
MC3	[I,4,SST]	113	MC3	1	Mc3	[I,4,SST]	113	C3	WMc	5.02	SMcC
MC4	[I,4,ZMF]	114	MC4	1	Mc4	[I,4,ZMF]	114	C4	WMc	5.0	SMcD
MC5	[I,6,ABA]	121	MC5	2	Mc5	[I,6,ABA]	121	C5	WMc	4.7	SMcE
MC6	[I,6,BFT]	122	MC6	2	Mc6	[I,6,BFT]	122	C6	WMc	4.95	SMcF
MC7	[I,6,SST]	123	MC7	2	Mc7	[I,6,SST]	123	C7	WMc	5.32	SMcG
MC8	[I,6,ZMF]	124	MC8	2	Mc8	[I,6,ZMF]	124	C8	WMc	5.45	SMcH
C1	[I,4,ABA]	111	C1	1	Aca1	[I,4,ABA,I,I,CMA]	111111	CA1	WAca	5.18	SAcaA
TP1	[I]	1	TP1	1							
CM1	[I,CMA]	11	CM1	1							
C1	[I,4,ABA]	111	C1	1	Aca2	[I,4,ABA,0,I,CMB]	111012	CA2	WAca	5.01	SAcaB
TP2	[0]	0	/	/							
CM2	[I,CMB]	12	CM2	2							
...	...	...	...	...	...	...	...	...	...	...	...
C8	[I,6,ZMF]	124	C8	1	Aca30	[I,6,ZMF,I,I,CMB]	124112	CA30	WAca	5.65	SAcaD
TP1	[I]	1	TP1	1							
CM2	[I,CMB]	12	CM2	1							
...	...	...	...	...	...	...	...	...	...	...	...

4.5.3. Generic Planning

In general, the variety handler at the end product level (e.g., VM in Table 4.5) is the most comprehensive variety handler representing the composite of all the variety handlers involved in a process platform. The states of such an end variety handler (e.g., θ_{VM}^*) indicate all the feasible instantiations of variety parameters associated with generic items within the product and process families. For instance, a particular product variant (ξ) can be described as specifications of relevant variety parameters, as shown in Table 4.7. Applying the state specification algorithm, all variety handlers relevant to the product can be populated and obtain their valid values, as given in Table 4.8. The specification of the product can be represented as a specific state of the end variety handler:

$\theta_{VM\xi}^* = [I, WION, 2.5, I, 12, PVC, 0, I, CMB, I, 4, BFT, I, 4, II, I, 3.5, I, O, BAA, I, O, 4.5, I, TL, 0, *]$. For this

product, components TP and RH are not included in the BOM structure as the customer does not specify any requirement on the Tape and Rubber Holder (i.e., Tape.IncludeFlag=0 and RubberHolder.IncludeFlag=0). As the RH is not to be included (i.e., the cardinal flag of its variety handler RH.F=0), a ‘*’ is assigned to all other variety parameters of RH variety handler (i.e., RH.S=*), meaning any value can be assumed.

Table 4.7 Specification of a customized product

Customized Product: ξ	
Functional Requirement	Specification
Coil.Length	4mm
Coil.WindingMode	BFT
Frame.OuterDiameter	4mm
Frame.Length	11mm
BracketA.Shape	O
BracketA.Thickness	BAA
BracketB.Shape	O
BracketB.Width	4.5mm
Terminal.Pitch	TL
Commutator.Thickness	CMB
Shaft.Length	12mm
Shaft.Material	PVC
Magnet.OuterDiameter	3.5mm
Weigh.Radius	2.5mm
Weigh.Weight	WION

Table 4.8 States of variety handlers relevant to the product

$$\begin{aligned}
\Theta_{MT\xi}^* &= \Theta_{Qnt\xi}^* = \Theta_{T\xi}^* = \Theta_{Qt\xi}^* = \Theta_{W-Mt\xi}^* = \Theta_{T-Mt\xi}^* = \Theta_{S-Mt\xi}^* = [I, TL] \\
\Theta_{MBB\xi}^* &= \Theta_{Qmbb\xi}^* = \Theta_{BB\xi}^* = \Theta_{Qbb\xi}^* = \Theta_{W-Mbb\xi}^* = \Theta_{T-Mbb\xi}^* = \Theta_{S-Mbb\xi}^* = [I, O, 4.5] \\
\Theta_{MBA\xi}^* &= \Theta_{Qmba\xi}^* = \Theta_{BA\xi}^* = \Theta_{Qba\xi}^* = \Theta_{W-Mba\xi}^* = \Theta_{T-Mba\xi}^* = \Theta_{S-Mba\xi}^* = [I, O, BAA] \\
\Theta_{MF\xi}^* &= \Theta_{Qmf\xi}^* = \Theta_{F\xi}^* = \Theta_{Qf\xi}^* = \Theta_{W-Mf\xi}^* = \Theta_{T-Mf\xi}^* = \Theta_{S-Mf\xi}^* = [I, 4, 11] \\
\Theta_{MC\xi}^* &= \Theta_{Qmc\xi}^* = \Theta_{C\xi}^* = \Theta_{Qc\xi}^* = \Theta_{W-Mc\xi}^* = \Theta_{T-Mc\xi}^* = \Theta_{S-Mc\xi}^* = [I, 4, BFT] \\
\Theta_{TP\xi}^* &= \Theta_{Qtp\xi}^* = [0] \\
\Theta_{CM\xi}^* &= \Theta_{Qcm\xi}^* = [I, CMB] \\
\Theta_{S\xi}^* &= \Theta_{Qs\xi}^* = [I, 12, PVC] \\
\Theta_{CA\xi}^* &= \Theta_{Qca\xi}^* = \Theta_{W-Aca\xi}^* = \Theta_{T-Aca\xi}^* = \Theta_{S-Aca\xi}^* = [0, I, CMB, I, 4, BFT] \\
\Theta_{M\xi}^* &= \Theta_{Qm\xi}^* = [I, 3.5] \\
\Theta_{AA\xi}^* &= \Theta_{Qaa\xi}^* = \Theta_{W-Aaa\xi}^* = \Theta_{T-Aaa\xi}^* = \Theta_{S-Aaa\xi}^* = [I, 12, PVC, 0, I, CMB, I, 4, BFT] \\
\Theta_{FA\xi}^* &= \Theta_{Qfa\xi}^* = \Theta_{W-Afa\xi}^* = \Theta_{T-Afa\xi}^* = \Theta_{S-Afa\xi}^* = [I, 4, 11, I, 3.5] \\
\Theta_{AB\xi}^* &= \Theta_{Qab\xi}^* = \Theta_{W-Aba\xi}^* = \Theta_{T-Aba\xi}^* = \Theta_{S-Aba\xi}^* = [I, O, BAA, I, O, 4.5, I, TL] \\
\Theta_{W\xi}^* &= \Theta_{Qw\xi}^* = [I, WION, 2.5] \\
\Theta_{MB\xi}^* &= \Theta_{Qmb\xi}^* = \Theta_{W-Amb\xi}^* = \Theta_{T-Amb\xi}^* = \Theta_{S-Amb\xi}^* \\
&= [I, 12, PVC, 0, I, CMB, I, 4, BFT, I, 4, 11, I, 3.5, I, O, BAA, I, O, 4.5, I, TL] \\
\Theta_{RH\xi}^* &= \Theta_{Qrh\xi}^* = [0, *] \\
\Theta_{IM\xi}^* &= \Theta_{W-Avm\xi}^* = \Theta_{T-Avm\xi}^* = \Theta_{S-Avm\xi}^* \\
&= [I, WION, 2.5, I, 12, PVC, 0, I, CMB, I, 4, BFT, I, 4, 11, I, 3.5, I, O, BAA, I, O, 4.5, I, TL, 0, *]
\end{aligned}$$

Referring to the process variety grid constructed in Table 4.6, all valid variants corresponding to the valid variety handler states specified in Table 4.8 can be identified. Accordingly, the material, product and resource domains of valid operation variants become non-empty, as shown in Table 4.9. As TP.F=0 and RH.F=0, there is no valid TP or RH variant in $D_{Aca\xi}^M$ or $D_{Avm\xi}^M$.

Table 4.9 Valid operation variants represented in domains

Operation (O_{ξ}^*)	Material Domain ($D_{\xi}^M = \{(\theta_{\xi}^*, Co_{\xi}^*, Q_{\xi}^*) y=1, \dots, M_{\xi}\}$)	Product Domain ($D_{\xi}^P \equiv \{(\theta_{\xi}^*, Co_{\xi}^*, I)\}$)	Resource Domain ($D_{\xi}^R \equiv \{(\theta_{\xi}^*, W_{\xi}^*, T_{\xi}^*, S_{\xi}^*)\}$)
$Mt_{\xi}^* = Mt1$	$(\theta_{MT\xi}^*, MT1, 2)$	$(\theta_{T\xi}^*, T1, 1)$	$(\theta_{Mt\xi}^*, WMtA, 4.93, SMtA)$
$Mbb_{\xi}^* = Mbb1$	$(\theta_{MBB\xi}^*, MBB1, 2)$	$(\theta_{BB\xi}^*, BB1, 1)$	$(\theta_{Mbb\xi}^*, WMbba, 4.05, SMbbaA)$
$Mba_{\xi}^* = Mba1$	$(\theta_{MBA\xi}^*, MBA1, 3)$	$(\theta_{BA\xi}^*, BA1, 1)$	$(\theta_{Mba\xi}^*, WMbba, 5.25, SMbaA)$
$Mf_{\xi}^* = Mf5$	$(\theta_{MF\xi}^*, MF5, 4)$	$(\theta_{F\xi}^*, F5, 1)$	$(\theta_{Mf\xi}^*, WMfC, 5.07, SMfC)$
$Mc_{\xi}^* = Mc2$	$(\theta_{MC\xi}^*, MC2, 1)$	$(\theta_{C\xi}^*, C2, 1)$	$(\theta_{Mc\xi}^*, WMc, 4.78, SMCB)$
$Aca_{\xi}^* = Aca4$	$(\theta_{CA\xi}^*, CM2, 2), (\theta_{C\xi}^*, C2, 1)$	$(\theta_{CA\xi}^*, CA4, 1)$	$(\theta_{Aca\xi}^*, WAca, 5.18, SAcaD)$
$Aaa_{\xi}^* = Aaa3$	$(\theta_{SA\xi}^*, S3, 1), (\theta_{CA\xi}^*, CA4, 1)$	$(\theta_{AA\xi}^*, AA3, 1)$	$(\theta_{Aaa\xi}^*, WAaaB, 5.30, SAaaC)$
$Afa_{\xi}^* = Afa2$	$(\theta_{FA\xi}^*, F5, 1), (\theta_{MA\xi}^*, M1, 1)$	$(\theta_{FA\xi}^*, FA2, 1)$	$(\theta_{Afa\xi}^*, WAfa, 4.48, SAfaB)$
$Aba_{\xi}^* = Aba4$	$(\theta_{BA\xi}^*, BA1, 1), (\theta_{BB\xi}^*, BB1, 1), (\theta_{T\xi}^*, T1, 2)$	$(\theta_{AB\xi}^*, AB4, 1)$	$(\theta_{Aba\xi}^*, WAba, 4.04, SAbaB)$
$Amb_{\xi}^* = Amb3$	$(\theta_{AA\xi}^*, AA3, 1), (\theta_{FA\xi}^*, FA2, 1), (\theta_{AB\xi}^*, AB4, 1)$	$(\theta_{AB\xi}^*, MB3, 1)$	$(\theta_{Amb\xi}^*, WAmb, 9.25, SAmbC)$
$Avm_{\xi}^* = Avm2$	$(\theta_{W\xi}^*, W2, 1), (\theta_{MB\xi}^*, MB3, 1)$	$(\theta_{VM\xi}^*, VM2, 1)$	$(\theta_{Avm\xi}^*, WAvm, 8.78, SAvmB)$

Further applying the routing configuration algorithm to the domains of valid operation variants, a specific routing variant can be configured to meet the customized product ξ , i.e.,

$$\Omega_{\xi} = \{Mc2 \succ Aca4, Aca4 \succ Aaa3, Aaa3 \succ Amb3, Mf5 \succ Afa2, Afa2 \succ Amb3, Mba1 \succ Aba4, Mbb1 \succ Aba4, Mt1 \succ Aba4, Aba4 \succ Amb3, Amb3 \succ Avm2\} \quad (4.9)$$

The corresponding precedence graph is given in Figure 4.7.

Figure 4.7 A specific routing for the customized product

4.6. Summary

To provide a holistic view and the basic constructs of a process platform, process platform formulation is carried out from the architecture point of view. Process platform architecture conveys information about organization of various product and process data as well as configuration of routings. To establish rigorous process platform models, operations are used as the central points for connecting all the relevant product and process elements. The set of parameters and their value instances associated with both product and process data are employed to formulate routing configuration and variety coordination. This way describing a process platform provides a complete view of a process platform in terms of what (is a process platform), which (functionalities to be performed), and how (to achieve these functionalities), which in turn facilitates the development of solutions to address other issues involved in process platform-based production configuration. In the next chapter, process platform will be approached from the structural aspect, that is, structural representation.

CHAPTER 5:

STRUCTURAL REPRESENTATION OF PROCESS PLATFORM BASED ON UNIFIED MODELING LANGUAGE

This chapter presents the various constituent elements along with the complex relationships among them in a process platform. The elements and relationships reflect a process platform from the structural viewpoint, i.e., the syntax of a process platform. Understanding of process platform syntax is conducive to tackling all issues involved in process platform-based production configuration, e.g., process platform development, production configuration automation. This can be accomplished through analyzing the requirements and characteristics of the problems to be handled by referring to process platform structural models. The diverse elements of a process platform can be classified into different groups thus forming classes, each of which contains a number of members. Therefore, the complex relationships among the entities are embodied by class-member relationships, including class-to-class, class-to-member, and member-to-member. In this regard, the structural representation of a process platform calls for a proper modeling tool that can sufficiently address the various relationships and model each element accurately as well. Since the UML incorporates object-oriented technology, it is suitable to represent both the internal structures and external interrelations of the elements of a system to be modeled (Rumbaugh et al., 1991; Booch, 1994). This research thus employs UML to establish structural models of a process platform. In the next section representation languages are reviewed.

5.1. Representation Languages

A number of modeling tools have been developed attempting to accurately describe systems to be modeled. It is common that these tools capture the modeled systems from one

or more perspectives and reflect certain characteristics of systems from specific viewpoints. Integrated definition methods (IDEF), comprising IDEF0, IDEF1, IDEF1X (X is short for extended), IDEF2, IDEF3, IDEF4, and IDEF5 are commonly used in enterprise modeling ranging from functional modeling, information modeling, data modeling, process flow modeling, simulation design, OO design to ontological analysis and design; however, they are incapable of analyzing a system's structure (<http://www.idef.com>). Although IDEF1X is designed to model entities and relationships, essentially it is the extension of entity-relationship modeling scheme and cannot deal with complex data models. Data flow diagrams can capture the flow of data from external entities into the system and how the data moves from one process to another. They put more emphasis on the system dynamics that is embodied by data movement but are not suitable for describing the data structure of a system (<http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm>).

In recent years, UML has been adopted by Object Management Group (OMG) as an object-oriented modeling language with formal syntax and semantics (Rumbaugh et al., 1999). Moreover, UML specification (Version 1.4.2) has been accepted by ISO (International Organization Standard) as an ISO specification (<http://www.omg.org/>). OMG describes UML as *"A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems"*. UML can be used not only to model software systems but also to model non-software systems. Six out of thirteen types of diagrams defined by UML represent the static structure of a system. Among them, the class diagram can explicitly represent the elements along with their interconnections. In order to adapt this general modeling language to specific application domains, a standardized extension mechanism is defined.

Ever since the publication of UML by OMG, a number of researchers have investigated UML's applications, discussed its advantages and disadvantages, e.g., Nunes et al. (2004). Kogut et al. (2002) discussed the applications of UML for ontology representation and

development. Bauer and Odell (2005) analyzed the application issues of UML 2.0 for the specification of agent-based systems. Selonen et al. (2003) gave a thorough review and analysis of the transformation issues of UML's diagrams. They discussed in detail the most interesting transformation operations. Mellor (2003) provided some insight into XTUML (Executable and Translatable UML)—a subset of UML and described the behind ideas. Attempting to identify objects in the software systems to be developed, a systematic method using UML to build object models in embedded systems was proposed by Kimour and Meslati (2004). Harrison et al. (2000) proposed an approach to generating Java implementation code from UML diagrams that are used to describe models. In their approach, UML diagrams can be specified at a higher level than the current adopted tools so that fewer constraints will be put on the supported UML constructs, e.g., generalizations and association classes.

The UML class diagram is adopted in this chapter to model entities and relationships of process platforms. Furthermore, the standard extension mechanism is used to adapt UML to fit the representation needs. In the next section, the UML class diagram and the standard extension mechanism are introduced.

5.2. UML Constructs

5.2.1. Class Diagram

The metamodel of UML consists of thirteen diagrams that are tailored to specify distinguished aspects of a system to be modeled. These diagrams can be grouped into three categories including (1) structural diagrams, (2) behavioral diagram, and (3) interaction diagrams. UML allows specifying structural aspect of a system to be modeled in class diagrams. Class diagram consists of classes and various static relationships between them that together describe the structures of models, the elements that exist, their internal structures,

and their relationships to others. The metamodel of the UML class diagram is summarized and shown in Figure 5.1.

Figure 5.1 Metamodel of UML class diagram

Classes are drawn as rectangles, in which the top compartment stands for the name of the class, the middle one for the attributes and the bottom one for the operations. Both the attribute and operation compartments can be suppressed. The relationships between classes are drawn as annotated lines between their rectangles.

There are five types of relationships between classes, namely association, dependency, aggregation, generalization, and composition. An association reflects a binary relationship between two classes, and it can be drawn as a solid line, which is composed of one or more connected segments with some necessary adornments. The adornments include an association name with or without a small black solid triangle (the point of which indicates the direction to read the name), multiplicities indicating the cardinalities of instances of the attached classes, arrows and diamonds (optional) at the end of the solid line of the association, and rolenames (optional) of the attached classes. The association name designates the name of the association. Multiplicities on associations are written as a number at each end, with the number applying to instances of the attached class. It can be shown as a comma-separated sequence of integer intervals representing a range of integers in the format:

Lower bound..Upper bound.

The star character (*) can be used for the upper bound, denoting an unlimited upper bound. For example, “*” indicates 0 or more, “1..*” denotes one or many, and a single integer value means the integer range only contains the single integer value.

The other four relationships derived from association are (1) generalization, (2) aggregation (shared aggregation), (3) composition, and (4) dependency. A generalization is shown as a solid-line path from the specific class (sub-class) to the general class (super-class) with a small empty triangle on the general class end of the association. A solid filled diamond and a hollow diamond are placed on the end of composition and shared aggregation at the composite side, respectively. Both of them indicate a part-whole relation between two classes. While the part in a composition connection can only be a part of one composite, the part of a shared aggregation can be a part of a number of wholes. A dependency indicates that the class (client class) at the tail of the dashed arrow line usually with a keyword depends on the class (supplier class) at the arrowhead, i.e., a change to the client needs the change to the supplier.

In order to meet the requirements of modeling process platform structures, the class diagram is extended to include instances and links. Two additional relationships, including instance_of and link, thus appear in the extended class diagram. An instance_of represents the connection between a class and its instances by a dashed arrow line pointing to the class with the keyword <<instance_of>>. Several instances of the same class can share one keyword. A link is the connection between instances and denoted by a solid line with/without keyword.

5.2.2. Extension Mechanism

A standardized extension mechanism is provided to adapt this general modeling language to specific application domains. The standard way of introducing an extension to UML is to add a profile (a stereotyped package), which is defined as a set of stereotypes

(classes of metamodel elements), tagged values (keyword-value pairs), and constraints. The advantage of an extension mechanism is that a profile does not introduce any new concepts to the language but rather than specialize it for certain domains. Therefore, with the extension mechanism UML can fit the particular modeling needs of different problem domains while keeping its integrity. To represent a process platform in a more precise manner, the stereotype instead of tagged values and constraints are used. Some special sets of classes (e.g., resource type, constituent type), links (e.g., compatible, select), associations (e.g., support, perform), and dependency (e.g., succeed) are thus defined. The UML profile for the domain of process platforms consists of the following elements (not exhaustive), as shown in Table 5.1.

Table 5.1 UML Profile for process platform

Metamodel Class	Stereotype
Class	Constituent
Class	Property
Class	Resource
Association	Support
Association	Perform
Link	Compatible
Dependency	Succeed
...	...

Figure 5.2 shows how the profile for process platform structures is integrated into the four-layer architecture of UML. In the metamodel in the third layer (L3) the extensions for the data structure domain are defined. The actual process platform model in the second layer (L2) corresponds to the class diagram. The routing list in the first layer (L1) for a given customized product derived from the process platform can be seen as the instantiation of the model in L2.

Figure 5.2 Layered UML architecture

5.3. Structural Representation of Process Platform

Process platform structural representation is approached from three aspects, namely generic product structures, generic process structures and integrated generic routing structures for the correspondence between generic product and process structures.

5.3.1. Generic Product Structure

The generic product structure contains several entities, including items in general, end products, assemblies, parts, and raw materials in particular. These items along with the aforementioned three kinds of relationships (i.e., class-to-class, class-to-member, and member-to-member) form the generic product structure.

Entities

The entities documented in the generic product structure convey product data and information from different levels of abstraction starting with end products and ending with specific values of variety parameters. While the highest level, i.e., the most abstract level, contains an end product, several sets of value instances of variety parameters are located at the lowest levels.

(1) *Item*. An entity of the type *item* represents a set of one or more product variants, each of which is either a physical product or a concept. It can be the end product at the top level of the generic product structure. Also it can be an assembly or a part at any arbitrary

intermediate level or the lowest level, and raw material at the lowest level. While a specific item represents only one product variant, a generic item represents a class.

(2) *End product*. An entity of the type *end product* represents a set of one or more product variants at the top level of the generic product structure. Entities of the type *end product* are configured by specifying compatible items at lower levels according to the customer requirements. The determination of items that are compatible to one another is influenced by the design parameters, which are transformed from customer requirements.

(3) *Assembly*. An entity of the type *assembly* represents a set of one or more product variants at any arbitrary intermediate level of the generic product structure. They can be assembled together to form the higher level assemblies or end products. An assembly has at least two child items, which may be assemblies or parts.

(4) *Part*. An entity of the type *part* represents a set of one or more product variants at the lowest level or any arbitrary intermediate level of the generic product structure. They are usually used to form assemblies at higher levels or the end products at the top level. A *part* entity is either a type of in-house manufactured parts at the intermediate level or a type of purchased parts at the lowest level of the hierarchy.

(5) *Raw material*. An entity of the type *raw material* represents a set of one or more raw material variants at the lowest level of the generic product structure. Raw materials are machined or fabricated into parts at the higher levels. A raw material variant is defined exactly the same as the part to be manufactured.

(6) *Parameter*. An entity of the type *parameter* is a special kind of attribute (properties of a product) and relevant to variety. It represents a characteristic of an entity of the type *item*. A parameter describes an item independent from its context. For instance, the material of an item keeps unchanged no matter where the item is used. An item usually has a number of entities of the type *parameter*.

(7) *Value*. An entity of the type *value* represents an assignment of an entity of the type *parameter*. Every such parameter may have a number of values. A particular item has a specific value for each of its parameters. After each specific value for the corresponding parameter of a generic item is determined, an item variant is obtained.

(8) *Parameter value*. An entity of the type *parameter value* defines a binary relationship between an entity of the type *parameter* and an entity of the type *value*. It is the combination of the two different types of entities. A set of such combinations defining a unique item is termed as product specification.

Relationships

(1) *Instance_of*. An instance_of relationship shows the connection between an instance and its class. For example, *material* can be an instance of the class *parameter*; *red* is an instance of the class *value*.

Figure 5.3 Links between instances

(2) *Association*. An association denotes a relationship between two classes that involve connections between their instances. As shown in Figure 5.3(a), the connection between two subclasses P_iV (the set of value instances of the i-th parameter) and P_jV (the set of value

instances of the j -th parameter) of the super-class *parameter value* indicates certain relationships. The association relationship between two classes is reified by the links between their instances, which may be incompatible or compatible. The association between *parameter value* and *parameter* indicates a number of parameter values can be mapped onto the same parameter, whilst a one-to-one mapping relationship exists between *value* and *parameter value*.

(3) *Link*. A link represents the connection between two instances. It itself is an instance of an association. A link may exist in three situations. In the first situation, it connects two instances possessed by two different classes. In case one class is the composition of the other, the relationship between the two instances is “consist_of” or “a_part_of”. In Figure 5.3(b), the link between A_{jj}^* (the j -th variant of the j -th generic assembly) and EP_j^* (the j -th end product) indicates an “a_part_of” relationship. If the two classes do not have the composition relationship, the connection between the two instances is either compatible (*AND*) or incompatible (*XOR*). In Figure 5.3(b), A_{ij}^* (the j -th variant of the i -th generic assembly) and A_{jj}^* (the j -th variant of the j -th generic assembly) are incompatible, whilst A_{ii}^* (the i -th variant of the i -th generic assembly) and A_{jj}^* (the j -th variant of the j -th generic assembly) are compatible. In Figure 5.3(a), $P_iV_{ij}^*$ (the j -th value instance of the i -th parameter) and $P_iV_{ii}^*$ (the i -th value instance of the i -th parameter) are two instances of the subclass P_iV ; $P_jV_{jj}^*$ (the j -th value instance of the j -th parameter) and $P_jV_{ji}^*$ (the i -th value instance of the j -th parameter) are two instances of the subclass P_jV . While $P_iV_{ii}^*$ is compatible with $P_jV_{ji}^*$, $P_iV_{ii}^*$ and $P_jV_{jj}^*$ are incompatible. If $P_iV_{ii}^*$ stands for “square shape”; $P_jV_{jj}^*$ and $P_jV_{ji}^*$ for “wood material” and “steel material”, respectively. The implication is that no item variant with a square shape can have wood as its material, or when a square item is chosen, the other item with material wood at the same level of the generic product structure is not allowed to be

assembled with it to form the parent. The compatible relationship indicates the reverse condition. In the second situation, a link connects two instances of one class. The relationship between such instances is *XOR*, which means at one time only one instance can be chosen to represent the class to produce a parent item. In the third situation, a link exists between the instances of the entity types of *item* and *parameter value/parameter/value*, as shown in Figure 5.3(c). Such a link has a keyword **has**. It indicates that an item has certain characteristics.

(4) *Dependency*. A dependency indicates a semantic relationship between two classes. The dependency between classes *parameter* and *value* implies parameters must be bound to actual values in order to express certain meaning. After the binding, a new class *parameter value* is created.

(5) *Shared aggregation*. A shared aggregation is a special kind of aggregation. It represents the relationship between a part and a whole, where the part can be a part in a number of wholes. The shared aggregation between *part* and *assembly/part* and *end product/assembly* and *end product/assembly* and *end product* indicates that a number of assemblies (or end products) can share a set of common parts and/or assemblies.

(6) *Generalization*. A generalization is the taxonomic relationship between a more general class (the super-class) and a more specific class (the sub-class) that is fully consistent with the super-class and adds additional information. The connection between *end product/assembly/part* and *item* indicates a generalization connection. In Figure 5.3(b), the connection between End Product and Assembly_i/End Product and Assembly_j is a generalization.

Representation of Generic Product Structure

Figure 5.4 shows the UML model of the generic product structure. An item may be a raw material, a part, an assembly, or an end product in the product hierarchy. An end product

consists of parts and assemblies, each of which may contain child items at lower levels. An assembly may be formed by joining its child parts and/or assemblies. In the two sub-classes of the class *part*, a class of machined parts is in relation to a class of raw materials. One or more items can have a set of common parameter values. The differences among such items thus lie in the optional parameter values specific to each item.

Figure 5.4 Representation of the generic product structure

5.3.2. Generic Process Structure

Entities

Representing process elements included in the process family, entities in the generic process structure are described as follows.

(1) *Operation*. An entity of the type *operation* represents an aggregation of a set of sequenced manufacturing steps, which in turn indicate a production job. These manufacturing

steps are performed either by labor, or a machine, or the combination of both. An operation can transform an item in terms of physical shapes and/or chemical properties from one stage to another stage. A generic operation represents a number of similar operation variants. While a generic operation corresponds to a generic item, an operation variant of it is to fulfill a specific variant of the generic item. A generic operation can be described by its attributes, including the generic input materials, generic output product, generic cycle time, generic manufacturing resources such as a generic machine, tool, fixture, and labor.

(2) *Machine*. An entity of the type *machine* performs a part or the whole of an operation. In order to complete the operation, tools or fixtures may be used. The selection of machines is affected by parameters of items.

(3) *Tool*. An entity of the type *tool* is a support device for an entity of the type *machine* or *labor* to carry out an entity of the type *operation*. A particular tool may be shared by different entities of the type *machine* or *labor*.

(4) *Fixture*. Unlike an entity of the type *tool*, an entity of the type *fixture* is a fixed auxiliary device for an entity of the type *machine* or *labor* to execute an operation. Usually entities of the type *fixture* are fixed at a certain place.

(5) *Labor*. An entity of the type *labor* can be either human operators or robots. The task for a *labor* entity is to (a) carry out a part or the whole of one particular operation using tools and/or fixtures, and (b) execute the preparation activities, termed as setup, before the actual operation commences. In this research, labor refers to the number of labor, which coincides with manufacturing practice.

(6) *Setup*. An entity of the type *setup* represents a set of activities. It is necessary to perform these activities before an entity of the type *operation* begins. Examples of entities of the type *setup* are material preparation and changing of tools and fixtures.

(7) *Cycle time*. An entity of the type *cycle time* records the duration of an entity of the type *operation* from the starting point to the ending point. Each operation must have its own cycle time. However the cycle times for several different operations may be same.

(8) *Material handling system*. An entity of the type *material handling system (MHS)* consists of a set of one or more separate automated guided vehicles (AGVs), articulated arm robots, conveyors, gantry robots and buffers or the combination of some of them. Its main function is to transport materials and finished/semi-finished parts to the designated place for further operation. Usually an entity of the type *MHS* is shared by a number of entities of the type *machine*.

Relationships

(1) *Dependency*. The dependency relationship between operations reveals that the start of one operation depends on finishing of its preceding operation (given all needed manufacturing resources and materials are available and enough). A dependency can also be found between an *operation* class and a *setup* class.

(2) *Instance_of*. An *instance_of* relationship between entities in the generic process structure represents the connection between a process element class and its instances. For instance, an operation variant to produce an item variant is an instance of the generic operation in relation to the generic item of the item variant.

(3) *Generalization*. A generalization relationship shows the connection between the more general class and the more specific class of same type process elements. The relationship between a super-operation (machine) class and its sub-operation (machine) classes indicates a generalization.

(4) *Association*. An association indicates certain relationship between two classes of process elements. As shown in Figure 5.5(a), entities of the type *operation* are performed by entities of the type *machine* with the support of an entity of the type *MHS*, each operation has

a cycle time. The association in Figure 5.5(b) shows entities of the type *labor* do setups, use tools and fixtures, and control machines as well.

Figure 5.5 Association between process elements

(5) *Link*. A link in the generic process structure represents the connection between two instances of two generic process elements. The connection between an instance of *machine* /*labor* and an instance of *tool* or *fixture* describes a link. Such a link has a keyword **use**. It implies that in order to fulfill the specific operations, machines/labor may use tools and fixtures to complete the activities.

Representation of Generic Process Structure

Figure 5.6 shows the generic process structure represented using UML. Operations are performed either by machines or labor, which use tools and/or fixtures. The operations are executed according to certain sequence, which is determined by the characteristics of the end products. The required setup activities must be completed by entities of the type *labor* before its succeeding operation commences. Each operation with the aid of MHS has a cycle time.

Figure 5.6 Representation of the generic process structure

5.3.3. Generic Routing Structure

The generic routing structure is formed through linking the generic product structure with the generic process structure using the correspondence in between. The integration of product data with process data is accomplished by employing the set of variety parameters and their values that are used by both generic structures. Variety coordination in process platforms is thus enabled by the set of parameters and their value instances. In this regard, planning rules are defined to specify process elements for the corresponding product elements by using the set of parameters and their value instances.

Correspondence between the Generic Product and Process Structures

The correspondence between the two generic structures is embodied by a set of mapping relationships, which can be classified into *association* between classes and *link* between instances.

(1) *Association*. An association between a product element class and a process element class reflects certain meaningful connections. The connection between an *operation* class and an *item* class indicates that items pose requirements on operations. The connection between *parameter value* and *labor/machine/tool/fixture* implies that different parameter values may select the same or different labor/machines/tools/fixtures; however parameter values may not have an influence on the selection of machines, tools, fixtures, and labor. For instance, two item variants of a class with different colors, *color_red* and *color_black*, may require the same manufacturing resources. As opposed to this, in most situations different materials and different shapes may require different resources. Figure 5.7 shows the association between *parameter value* and *machine/labor/tool/fixture*.

Figure 5.7 Association between product and process entities

(2) *Link*. Connections between product element classes and process element classes are embodied by the specific links between their respective instances. As shown in Figure 5.8, $P_j V_{ji}^*$ (the *i*-th value instance of the *j*-th parameter) - an instance of *parameter value* (PV) class selects L_j^* (the *j*-th labor variant), F_i^* (the *i*-th fixture variant), T_i^* (the *i*-th tool variant), M_{ii}^* (the *i*-th machine of the *i*-th type), and M_{ji}^* (the *i*-th machine of the *j*-th type).

Figure 5.8 Links between product and process entities

Representation of Generic Routing Structure

Figure 5.9 shows the UML model of the generic routing structure that underpins a process platform. Within a process platform, operations are specified according to the properties of the given product items. Parameter values of product items may not have an influence on the selection of the tools, fixtures, machines, and labor. Manufacturing resources collaborate to complete the specified operation. The completion of operations leads to new items produced. The MHS transports the produced items or raw materials to the next operation, and thus keeps the production moving smoothly.

5.4. Case Study

UML has been applied to the vibration motors in the case study of Chapter 4 in order to clearly provide the process platform structural model of vibration motors.

5.4.1. Product and Process Information

The main parts of a vibration motor are rubber holder, weight, and mainbody. The mainbody in turn consists of armature assy, bracket assy, and frame assy. Figure 5.10 shows a motor and its BOM structure. Table 5.2 lists the motor's BOM data records.

Figure 5.9 Representation of entities and relationships in a process platform

Figure 5.10 Vibration motor and its BOM structure

Table 5.2 BOM Data for vibration motor

Hierarchy Level	Parent Item	Component Item	Quantity Per
1	Vibration Motor	Mainbody	1
1	Vibration Motor	Rubber Holder	1
1	Vibration Motor	Weight	1
2	Mainbody	Frame Assy	1
2	Mainbody	Armature Assy	1
2	Mainbody	Bracket Assy	1
3	Bracket Assy	Bracket A	1
3	Bracket Assy	Bracket B	1
3	Bracket Assy	Terminal	1
3	Armature Assy	Shaft	1
3	Armature Assy	Coil Assy	1
3	Frame Assy	Frame	1
3	Frame Assy	Magnet	1
4	Coil Assy	Coil	1
4	Coil Assy	Tape	1
4	Coil Assy	Commutator	1

The routing for a vibration motor involves 6 assembly operations (A_{vm} , A_{mb} , A_{aa} , A_{ca} , A_{fa} , A_{ba}) and 5 manufacturing operations (M_t , M_{ba} , M_{bb} , M_f , M_c), which may be shaping operation, machining operation, stamping operation, forming operation, and other operation types. Table 5.3 gives process data and information of the motor in Figure 5.10.

5.4.2. Representation of the Motor Process Platform

All of the parts of vibration motors can be customized to meet the individual customer requirements; thus, they are defined as generic items. Each of these generic items has a number of options providing possible selections to customers. The customer can select the specific features and options to build their expected motors. As a result, the feasible

enormous customized vibration motors cause numerous different routings on the shop floor. The motor process platform has been constructed *a priori*. Figure 5.11 shows the generic product structure for bracket assy (BAssy), which is composed of a bracket A (BA), bracket B (BB) and terminal (TL).

Table 5.3 Process data of vibration motor's routing

Sequence no.	Operation	Machine (Mc.)	Tool	Fixture	Labor No./Mc.	Cycletime Sec./item
50	Vibration Motor assembly	Wcaulking Mc..	...	Wsetting jig & Wcaulking head	1	9.25
40	Mainbody assembly	Fcaulking Mc. & Ainserting Mc.	...	Caulking blade & Bracket holder	1	9.25
30	Armature assembly	Sinserting Mc. & Soldering Mc.	...	Supporting holder & Pallet	1	5.30
20	Coil assembly	...	...	Guiding jig	1	5.18
10	Coil fabrication	Cwinding Mc.	...	Tray	1	4.52
20	Frame assembly	FMpressing Mc.	...	Fholder & Mholder	1	4.48
10	Frame fabrication	Fstamping Mc.	Die	Holder	1	5.07
20	Bracket assembly	Bfusing Mc.	Binserter	Bpressing jig	1	4.04
10	Bracket A fabrication	Binjection Mc.	Badjustor	Blocator	1	5.25
10	Bracket B fabrication	Binjection Mc.	Badjustor	Bprealignment jig	1	5.25
10	Terminal fabrication	Tcutting Mc.	Die	Tholder	1	4.93

* Same sequence numbers indicate parallel operations.

In the figure, “BAssy (I_0)”, “Bracket A (I_1)”, “Bracket B (I_2)”, and “Terminal (I_3)” indicate that BAssy, BA, BB, and TL are the 0-th, 1-th, 2-th, and 3-th generic items. “ BA_{li}^* ”, “ BB_{2k}^* ” and “ TL_{3j}^* ” denote the i -th BA variant, the k -th BB variant and the j -th TL variant. The parameters of generic items are represented as “parameter name (P_{xy})” in the figure, where x and y are indices of the generic items and the parameters of the generic items, such that “Color (P_{1l})” means the first parameter of the first generic item BA; and “Shape (P_{3l})” denotes the first parameter of the third generic item TL. Each parameter has several value instances. For example, “ L ” is coded for a particular shape of TL variants, and there are three different shapes, namely “ L ”, “ T ” and “ U ”. All parameters are bound to their possible value instances, and thus forms parameter values ($P_{xy}V_{xyz}^*$), where x , y and z are indices of the

generic items, parameters and value instances, such that $P_{xy}V_{xyz}^*$ indicates the z-th value of the y-th parameter of the x-th generic item.

Figure 5.11 UML-based representation of the generic product structure of a motor process platform

In Figure 5.11, the AND and XOR links show the compatibility and incompatibility of parameter values. Among the specific parameter values of BA variants, $P_{12}V_{123}^*$ (shape L) is compatible with $P_{13}V_{132}^*$ (width 6mm). It means the width of a particular L shape BA variant can be 6mm.

Table 5.4 shows the description of the generic operations involved in producing BAssy variants. All the value instances for generic process elements are written as X_{xy}^* , wherein X denotes the abbreviated name of generic process element; x and y denotes the indices of the generic items in relation with the generic process elements and the instances of generic process elements, respectively. For example, M_{11}^* describes a particular machine for

fabricating BA variants. Figure 5.12 illustrates the generic process structure of BA fabrication. The symbols in this figure are described in Table 5.4. Integrating BA's process data with product data forms the unified generic routing structure for fabricating BA variants, as shown in Figure 5.13. The specific values of the symbols are given in Table 5.4. For example, to manufacture a particular BA variant with a “U” shape, the injection machine Mc04 must be used. The resulted cycle time is 5.25 second per unit.

Table 5.4 Description of generic operations for Bracket Assy

Generic Operation{O}	Generic Item{Ii}	Variety Parameter {Pi}	Variety Value{V _y [*] _{ik} }	Variety PV{PV _y [*] _{ik} }	Variety Machine{M _{ik} [*] }	Variety Tool{T _{ik} [*] }	Variety Fixture{F _{ik} [*] }	Variety Labor{L _{ik} [*] }	Variety Cycletime{CT _{ik} [*] }
Bracket A Fabrication (Mba)	Bracket A {I1}	Color(P11)	Blue (V ₁₁₁ [*]) Red (V ₁₁₂ [*])	P ₁₁ V ₁₁₁ [*] =C_B P ₁₁ V ₁₁₂ [*] =C_R	M ₁₁ [*] =Binjection Mc06 M ₁₂ [*] =Binjection Mc04	T ₁₁ [*] =BadjustorA T ₁₂ [*] =BadjustorVII	F ₁₁ [*] =BlocatorA F ₁₂ [*] =BlocatorII F ₁₃ [*] =BlocatorIII	L ₁₁ [*] =1 L ₁₂ [*] =2	CT ₁₁ [*] =5.50 CT ₁₂ [*] =5.25
		Shape(P12)	U (V ₁₂₁ [*]) T (V ₁₂₂ [*]) L (V ₁₂₃ [*])	P ₁₂ V ₁₂₁ [*] =S_T P ₁₂ V ₁₂₂ [*] =S_U P ₁₂ V ₁₂₃ [*] =S_L					
		Width(P13)	5 (V ₁₃₁ [*]) 6 (V ₁₃₂ [*]) 7 (V ₁₃₃ [*])	P ₁₃ V ₁₃₁ [*] =W_5 P ₁₃ V ₁₃₂ [*] =W_6 P ₁₃ V ₁₃₃ [*] =W_7					
		* unit: mm							
Bracket B Fabrication (Mbb)	Bracket B {I2}	Color(P21)	Blue (V ₂₁₁ [*]) Red (V ₂₁₂ [*])	P ₂₁ V ₂₁₁ [*] =C_B P ₂₁ V ₂₁₂ [*] =C_R	M ₂₁ [*] =Binjection Mc06 M ₂₂ [*] =Binjection Mc04	T ₂₁ [*] =BadjustorA T ₂₂ [*] =BadjustorVII	F ₂₁ [*] =Bprealignment jigI F ₂₂ [*] =Bprealignment jigII F ₂₃ [*] =Bprealignment jigIII	L ₂₁ [*] =1 L ₂₂ [*] =2	CT ₂₁ [*] =5.25 CT ₂₂ [*] =6.04
		Shape(P22)	U (V ₂₂₁ [*]) T (V ₂₂₂ [*]) L (V ₂₂₃ [*])	P ₂₂ V ₂₂₁ [*] =S_T P ₂₂ V ₂₂₂ [*] =S_U P ₂₂ V ₂₂₃ [*] =S_L					
		Width(P23)	5 (V ₂₃₁ [*]) 6 (V ₂₃₂ [*]) 7 (V ₂₃₃ [*])	P ₂₃ V ₂₃₁ [*] =W_5 P ₂₃ V ₂₃₂ [*] =W_6 P ₂₃ V ₂₃₃ [*] =W_7					
		* unit: mm							
Terminal Fabrication (Mf)	Terminal {I3}	Shape(P31)	02 (V ₃₁₁ [*]) 11 (V ₃₁₂ [*])	P ₃₁ V ₃₁₁ [*] =S_02 P ₃₁ V ₃₁₂ [*] =S_11	M ₃₁ [*] =Tcutting McI M ₃₂ [*] =Tcutting McII	T ₃₁ [*] =DieLS T ₃₂ [*] =DieII	F ₃₁ [*] =TholderI F ₃₂ [*] =TholderII F ₃₃ [*] =TholderIII	L ₃₁ [*] =1 L ₃₂ [*] =2	CT ₃₁ [*] =4.93 CT ₃₂ [*] =4.45
		Width(P32)	5 (V ₃₂₁ [*]) 7 (V ₃₂₂ [*])	P ₃₂ V ₃₂₁ [*] =W_5 P ₃₂ V ₃₂₂ [*] =W_7					
		* unit: mm							
Bracket Assembly (Aba)	Bracket Assy {I0}				M ₀₁ [*] =Bfusing McI M ₀₂ [*] =Bfusing McII	T ₀₁ [*] =BinsertorV T ₀₂ [*] =BinsertorS	F ₀₁ [*] =Bpressing jigAI F ₀₂ [*] =Bpressing jigBO	L ₀₁ [*] =1 L ₀₂ [*] =2	CT ₀₁ [*] =4.04 CT ₀₂ [*] =4.25
								* no./machine	* sec./item

Figure 5.12 A part of the generic process structure of a motor process platform

The set of variety parameters and their value instances of motors' product items are used to select the proper process elements. For a given vibration motor, Table 5.5 gives the configuration result from selecting proper M (machine), T (tool), F (fixture), L (labor), and CT (cycle time) according to its specific BAssy variant (shown in Figure 5.14). Table 5.6 gives the complete routing configuration for this motor. Also indicated in the figure is that different colors have no impact on the selection of manufacturing resources.

Figure 5.13 Representation of a motor process platform (for Bracket A)

Figure 5.14 A variant of Bracket assy in a particular motor

Table 5.5 Specific routing data configured from the process platform in Figure 5.13

Sequence no.	Operation	Machine	Tool	Fixture	Labor no./mc	Cycletime sec./item	Material Item	Product Item	Quantity per
20	Bracket assembly	Bfusing McII	BinserterV	Bpressing jigAI	1	4.04	Bracket A Bracket B Terminal	Bracket Assy	1 1 1
10	Bracket A fabrication	Binjection Mc04	BadjustorA	BlocatorA	1	5.25	Raw Material	Bracket A	N/A
10	Bracket B fabrication	Binjection Mc04	BadjustorA	Bprealignment jigI	1	5.25		Bracket B	N/A
10	Terminal fabrication	Tcutting McI	DieII	TholderIII	1	4.93		Terminal	N/A
* Same sequence numbers indicate parallel operations									

Table 5.6 The complete routing data for the motor variant in Figure 5.14

Sequence no.	Operation	Machine	Tool	Fixture	Labor no./wc	Cycletime sec./item	Material Item	Product Item	Quantity per
50	Vibration Motor assembly	Wcaulking McI		Wsitting jigI Wcaulking headI	1	9.25	Weight Rubber Holder MainbodyAssy	Vibration Motor	1 1 1
40	Mainbody assembly	FcaulkingMcHS Ainserting McV		Caulking bladeII Bracket holderL	1	9.25	Armature Assy Frame Assy Bracket Assy	Mainbody Assy	1 1 1
30	Armature assembly	Sinserting and Soldering Mc.		Supporting holderI PalletII	1	5.30	Coil Assy Shaft	Armature Assy	1 1
20	Coil assembly			Guiding jigIII	1	5.18	Coil Tape Commuter	Coil Assy	1 1 1
30	Coil fabrication	Cwinding McV5		TrayII	1	4.52	Raw Material	Coil	N/A
20	Frame assembly	FMpressing Mc.		FholderI MholderVII	1	4.48	Magnet Frame	Frame Assy	1 1
10	Frame fabrication	Fstamping McI	Die	Holder01	1	5.07	Raw Material	Frame	N/A
20	Bracket assembly	Bfusing McII.	BinserterV	Bpressing jigAI	1	4.04	Bracket A Bracket B Terminal	Bracket Assy	1 1 1
10	Bracket A fabrication	Binjection Mc04	BadjustorA	BlocatorA	1	5.25	Raw Material	Bracket A	N/A
10	Bracket B fabrication	Binjection Mc04	BadjustorA	Bprealignment jigI	1	5.25		Bracket B	N/A
10	Terminal fabrication	Tcutting McI	DieII	TholderIII	1	4.93		Terminal	N/A
* Same sequence numbers indicate parallel operations									

5.5. Summary

In this chapter, process platforms are analyzed with respect to constituent elements and relationships. This is to clarify structures of process platforms. With consideration of the various elements and the multiple class-member relationships among them, UML is adopted in this research to accurately build process platform structural models. Process platform structural representation is accomplished through modeling the generic product structure, the generic process structure and correspondence in between. Integrating the generic product and process structures using the correspondence in between forms the generic routing structure that underpins a process platform. The reported case study of vibration motors shows the

appropriateness of UML for representing process platform structures and the process platform structural models. In the next chapter, a data mining methodology will be introduced for identifying generic routings from existing massive data.

CHAPTER 6:

GENERIC ROUTING IDENTIFICATION BASED ON DATA MINING

This chapter presents a data mining methodology to identify generic routings for process families and to support process platform construction. Since generic routing structures underpin process platforms, process platform construction can thus be facilitated by forming the generic routings of the associated process families. In a company's databases, there exists a large body of historical product and process data, behind which hidden patterns of common routings represent useful product and process knowledge. The reuse of existing product and process knowledge suggests itself as a natural technique to facilitate the handling of process variety and tradeoffs between design changes and process variations. As claimed by Chen et al. (1996), data mining has been well recognized as an efficient tool to uncover knowledge, i.e., previously unknown and potentially useful patterns of information, from existing data. Therefore, a systematic data mining methodology is developed in this research to identify generic routings from historical data. The next section reviews the related work about data mining techniques, including data mining, text mining and tree matching.

6.1. Data Mining Techniques

6.1.1. Data Mining

Data mining emerged during the late 1980's as the need for new technologies and automated tools that can intelligently assist people in transforming the vast amounts of data into useful information and knowledge has been generated. It is a multidisciplinary field, drawing work from areas including database technology, artificial intelligence, machine learning, neural networks, statistics, pattern recognition, knowledge-based systems,

knowledge acquisition, information retrieval, high-performance computing, and data visualization. Anand and Buchner (1998) defined data mining as the discovery of non-trivial, implicit, previously unknown, and potentially useful and understandable patterns from large data sets. Some authors treat data mining as a synonym for another popularly used term, knowledge discovery in databases. Alternatively, others view data mining as simply an essential step in the process of knowledge discovery in databases. Han and Kamber (2001) described several steps involved in knowledge discovery as data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation, and knowledge presentation. They viewed data mining as a step of the knowledge discovery process.

Westphal and Blaxton (1998) identified four functions of data mining including classification, estimation, segmentation, and description. Classification involves assigning labels to previously unseen data records based on the knowledge extracted from historical data. Estimation is the task of filling in missing values in the fields of an incoming record as a function of fields in other records. Segmentation (also called clustering) divides a population into smaller subpopulations with similar behavior. Clustering methods maximize homogeneity within a group and maximize heterogeneity between the groups. The description task focuses on explaining the relationships among the data.

Berry and Linoff (1997) presented many examples and applications of data mining in marketing, sales, and customer support. More specifically in engineering, many applications have been managed. Buchner et al. (1997) described the concepts of data mining and their synergy with manufacturing environments. A generic process is introduced and applications are shown. In engineering design, Leu et al. (2001) presented a data mining approach for the prediction of tunnel support stability. This approach is used to determine the size of the tunnel support knowing rock type and construction parameters. Kusiak (2000) used data mining to extract rules for making predictions in the semiconductor industry. Numerous methods for the decomposition of data sets were discussed in Kusiak (2001). The

decomposition is used to enhance the quality of knowledge extracted from large databases by simplifying the data mining task. This decomposition is used for the prediction and prevention of manufacturing faults in wafers. Gertosio and Dussauchoy (2004) applied data mining to extract knowledge from an industrial truck manufacturer database. The goal was to discover knowledge in the data of the test engine process in order to significantly reduce the processing time. Agard and Kusiak (2004) introduced a data mining-based methodology for product family design. In the methodology, the clustering technique was used to identify customer segmentation first, and then based on the segmentation, correlations between customer requirements and design concepts were analyzed and extracted. Romanowski and Nagi (2004) developed a data mining approach to forming generic bills of materials with an attempt to support variant design activities, e.g., searching similar design.

6.1.2. Text Mining

Text mining, also known as text data mining (Hearst, 1997) or knowledge discovery from textual databases (Feldman and Dagan, 1995), refers generally to the process of extracting interesting and non-trivial patterns or knowledge from unstructured text documents. It can be viewed as a leap from data mining in structured databases. Text mining is a more complex task than data mining as it involves dealing with text data, which are inherently unstructured and fuzzy.

Visa (2001) described important steps in text mining: 1) data preprocessing with the key idea of transforming the textual data into the form that can be processed, e.g., summarizing long documents, 2) text encoding dealing with the representation of text documents, 3) object clustering handling the partition of a set of objects into groups, 4) information extraction (or text segmentation) concerning the break of documents into topically coherent multi-paragraph sub parts, and finally 5) information and knowledge distillation. Tan (1999) divided text mining into two broad areas: text refinement and knowledge distillation. Text

refinement refers to the process of converting a non-determined document format to a predetermined intermediate form (IF). Knowledge distillation aims at deducing hidden patterns for knowledge from IFs. Nasukawa and Nagano (2001) viewed the entire text mining procedure as three main parts, namely concept extraction from documents, knowledge extraction from concepts by using statistical analysis methods, and information and knowledge display. They defined concepts as simple keywords or features, which are used to compose a document. Weng and Liu (2004) summarized text mining from the above work as 1) converting a document to a format that can be processed later, 2) data exploration, and finally 3) the results display using an interface.

Missikoff (2003) discussed the adoption of text mining in supporting ontology building with the case of *SymOntos* (an ontology management system). Focusing on document management, Weng and Liu (2003) proposed a multiple concept mechanism to search similar documents using text mining. Yang and Lee (2004) introduced the application of text mining techniques to a corpus of Web pages attempting to automatically create Web directions and organize them into hierarchies. Incorporating multiple concepts into text classification techniques that are traditionally single-concept based, Weng and Liu (2004) developed an e-mail reply template in order to lower the burden of customer service personnel in answering e-mails.

6.1.3. Tree Matching

Tree matching has been recognized as one of the major concerns in pattern recognition, e.g., image processing (Samet, 1982) and natural language processing (Neff and Boguraev, 1989). In most work, trees are compared by finding the minimal cost sequence of tree editing operations - node deletion, node insertion, and node substitution - required to transform one tree to another. The minimal edition cost determines the distance of two trees and thus the similarity.

Some researchers have studied the similarity measure of ordered trees (i.e., trees in which the left-to-right order among siblings is fixed), for example, Wang et al. (1991), Sastry and Ranganathan (1995), Tai (1979). Researchers proposed their tree matching algorithms with an attempt to improve computational efficiency and the performance of algorithms proposed by others (Tai, 1979; Hoffmann and O'Donnell, 1982). Some systems have been developed using ordered tree matching algorithms to assist people in retrieving similar trees, for example, Wang et al. (1994).

In addition to ordered tree matching, studies also have been conducted on unordered tree matching so as to solve pattern recognition problems of which ordered trees are not proper representations such as Shasha et al. (1994), Ferraro and Godin (2003), and Romanowski and Nagi (2005). Ordered tree matching is easier than unordered matching because the order of sibling nodes is fixed. Same as the algorithms for matching ordered trees, algorithms for matching unordered trees used the tree editing operations to determine the distance between two trees. However, the common inefficiency of both kinds of algorithms have been observed by Valiente (2002) as it is practically impossible to enumerate all the transformation ways (due to non-restriction on the order of editing operations), and thus the optimal distance between two trees. Accordingly, he introduced the tree edit graph to indirectly transform one tree to another.

6.2. Generic Routing Identification

Based on the process platform formulation introduced in Chapter 4, a generic routing GR can be defined as a tuple: $GR = \langle A, \succ \rangle$, where the precedence relation $\succ$ describes the ordering of pairs of operations from the set $A = \{O_1, O_2, \dots, O_N\}$, and $O_i|_{i=1, \dots, N}$ denotes an individual operation class, may it be a type of manufacturing or assembly operation. Associated with each operation, $O \in A$, is an operation description, i.e., specification, such

that, $\Psi_O = \langle \Phi_O^M, \Phi_O^P, R_O \rangle$, where Φ_O^M , Φ_O^P and R_O are called the material, product and resource classes of operation O , respectively. The material flow of each operation is distinguished by component classes. A component class may be a type of end product, assembly, part or raw material. A material class of an operation is a set of component classes. It refers to the types of input materials that will be transformed to their parent class by the operation. A product class of an operation is also one of component classes, but it denotes one type of product that will be produced by the operation. Usually an operation produces only one type of product, but it may require multiple types of materials.

While Φ_O^M and Φ_O^P constitute the material flow through operation O , a resource class, R_O , characterizes the types of manufacturing resources required by the operation in order to produce Φ_O^P . A resource class of an operation, $O \in \Lambda$, is defined as a triplet: $R_O = \langle M_O, T_O, S_O \rangle$, where M_O , T_O and S_O stand for the machine, cycle time and setup classes that are consumed by the operation, respectively. Figure 6.1 illustrates multiple classes and their interrelationships within a GR.

Figure 6.1 Operations and other relevant classes within a generic routing

The definition above allows GRs to be represented by precedence graphs, where in both product, process data and information are documented. In real manufacturing, when an assembly operation takes place, each time only two material components are assembled

together. The two material components are the primary input instead of bolts, nuts, connectors and the like. Hence, a routing can be represented as a labeled binary tree, in which the maximal number of child nodes for each parent node is 2. A routing binary tree as shown in Figure 6.2 contains two types of process information – operations details and sequences. While details of an operation such as the product, materials, and manufacturing resources are embedded in the nodes of the representation tree, the operations precedence (i.e., the order of operations execution) is exhibited by the tree structure.

Figure 6.2 A binary tree representation of a routing

Two types of nodes are involved in a routing representation tree: a leaf node (noted as *l*-node) and an intermediate node (noted as *i*-node). Each *l*-node represents a manufacturing or assembly operation that consumes at least one primitive component (i.e., purchased parts and raw materials) to produce a compound component (i.e., assemblies and machined parts) required by downstream operations. Each *i*-node indicates an assembly operation that produces an assembly or end product. The materials of an *i*-node are all compound components, i.e., the assemblies or parts produced from prior operations (*l*-nodes or other *i*-nodes). For example in Figure 6.2, operations M1, M2, M3 and A4 are *l*-nodes, whilst operations A1, A2 and A3 are *i*-nodes.

6.3. Methodology

The mining of GRs from existing routings involves two types of data: (1) textual data associated with the nodes describing characteristics of individual operations, and (2) structural data associated with the arcs depicting precedence relationships among operations.

To solve this unique data mining problem, both the text mining and tree matching techniques are incorporated in the proposed methodology. The methodology encompasses three sequential stages, namely (1) routing similarity measure, (2) routing clustering, and (3) routing unification.

6.3.1. Routing Similarity Measure

Given a set of routings, $\Omega = \{R_1, \dots, R_P\}$, the similarity, S_{rs} , between two routings, R_r and R_s , is derived based on pairwise comparisons of P individual routings. In accordance with the textual and structural data associated with each routing, S_{rs} comprise two respective components: (1) node content similarity, S_{rs}^{NC} , and (2) tree structure similarity, S_{rs}^{TS} .

6.3.1.1. Node Content Similarity

The contents of a routing are embodied in the nodes of the binary representation tree. While two routings may be similar in their tree structures (operations sequences), they may be discerned by diverse node contents (operations characteristics). Node content similarity measures the degree of approximation of two routings in terms of their operations' descriptions. To cope with such textual data, the text mining technique is employed, which performs a sentence semantic analysis using natural language processing technologies (Atkinson-Abutridy et al., 2003).

For a given set of routings, there are a number of operation types (nodes), $\{O_j\}_N$, from which each individual routing is configured. Some routings may not assume all these operations, that is, they may comprise a subset of $\{O_j\}_N$.

Corresponding to P number of routings, each operation, $O_j \in \{O_j\}_N$, assumes a maximal number of P specific variants, $\{O_{jk}^*\}_{N \times P}$. Let $S_{rs}^{O_j}$ be the similarity measure of two operations variants of the j-th type, O_{jr}^* and O_{js}^* , $\forall r, s = 1, \dots, P$ and $r \neq s$, corresponding two routings,

R_r and R_s , respectively. The node content similarity between these two routings is thus given as the sum of their operations similarity measures, as the following:

$$S_{rs}^{NC} = \sum_{j=1}^N S_{rs}^{O_j} . \quad (6.1)$$

As described in Section 6.2, each operation description includes three aspects: materials, $\{\Phi_{O_j}^M\}$, the product, $\{\Phi_{O_j}^P\}$, and resources, $\{R_{O_j}\}$. Accordingly, the operation similarity measure, $S_{rs}^{O_j}$, consists of three elements, namely material similarity, $S_{rs}^{M_j}$, product similarity, $S_{rs}^{P_j}$, and resource similarity, $S_{rs}^{R_j}$. Therefore, the operation similarity measure is given as the following:

$$S_{rs}^{O_j} = S_{rs}^{M_j} + S_{rs}^{P_j} + S_{rs}^{R_j} . \quad (6.2)$$

Step 1: Material Similarity Measure

The basic construct of an operation description is shown in Figure 6.3. The materials of an operation are a set of components, which may be types of raw materials, parts, and/or assemblies, i.e., $\Phi_{O_j}^M = \{Co_{jk}^M \mid \forall k = 1, \dots, K_j\}$. For a labeled binary tree, there are always two material components, i.e., $K_j = 2$. As shown in Figure 6.3, operation variant, O, requires two material components, B and C. Similarly, another operation variant, O', consumes two material components, B' and C'. Therefore, material similarity of two operation variants, O_{jr}^* and O_{js}^* , is calculated based on all their material components. Since some material components may be more important than others for an operation, a weighted sum of individual material component similarity measures should be used, i.e.,

$$S_{rs}^{M_j} = \sum_{k=1}^{K_j} \left(w_{jk}^M S_{rs}^{Co_{jk}^M} \right), \quad (6.3)$$

where $S_{rs}^{Co_{jk}^M}$ refers to the component similarity measure between two component variants,

Co_{jkr}^{M*} and Co_{jks}^{M*} , and w_{jk}^M with $\sum_{k=1}^{K_j} w_{jk}^M = 1$ indicates the relative importance of $\{Co_{jk}^M\}_{K_j}$ in

regard to O_j . Such weight values can be determined based on domain knowledge. Component similarity indicates how similar two components are and to what extent they can be used as an alternative to the other.

Figure 6.3 Basic construct of an operation description and bipartite matching

For an l -node of a routing representation tree, the corresponding operation takes at least one primitive component as its input. If an operation is an i -node, its material components are all compound components that are produced from prior operations. Accordingly, the material similarity measure of two component variants, Co_{jkr}^{M*} and Co_{jks}^{M*} , depends on two cases: if component Co_{jk}^M is a primitive or a compound component. While a procedure for measuring primitive component similarity using text mining is devised, bipartite matching in graph comparison is adopted to measure compound component similarity.

Similarity measure for primitive components

(1) *Prepare data files.* Descriptions of all nodes are extracted from P number of routing representation trees and are organized into files according to their corresponding operations, i.e., each file keeps the records of the same type operation. For each file, component descriptions are further extracted and saved in a new file. Thus, corresponding to the number N of operation types in the P number of routings, there is the same number of N files, each of which is for component descriptions of the same type operation. Files for primitive components and compound components can be distinguished according to the associated operations. Similarity measure for primitive components proceeds as follows.

(2) *Encode semantics.* The primitive component data files must be organized into a proper format for text mining to work. A component is generally depicted by some

characteristics – more specifically a list of attribute values with respect to some descriptive fields. The basic attribute field is the name or ID of the component type. Figure 6.4 shows an example of attribute descriptions for a specific bracket variant used in vibration motors. In the figure, four attribute fields are used to describe a bracket component type, including “name/ID”, “shape”, “color”, and “diameter”. Different bracket variants assume different values of these attribute fields.

Two types of attributes can be distinguished: nominal and numerical. While a nominal attribute value is in the form of symbolic text, values of numerical attributes are numbers. In practice, a nominal attribute value itself is meaningful enough for identifying a unique record, e.g., “square” indicates the shape attribute. However, this is not the case for numerical attributes. A specific numerical value alone can not suggest which attribute field it pertains to. For example, “10mm” can indicate an instance of either “length” or “width”. Therefore, rather than listing single values, numerical attributes are described using attribute-value pairs, for example, “diameter12mm” in Figure 6.4.

Component Description
BRACKET A, SQUARE, BLACK, DIAMETER12mm

Figure 6.4 Component descriptions

(3) *Extract keywords.* A parser is used to scan a text file for primitive components. The result is a list of extracted significant words or phrases. The keywords are generated as separated records in three forms. Single words and word combinations constitute phrases, representing the values of nominal attributes. Word-number pairs are related to numerical attribute fields.

(4) *Derive occurrence frequencies.* All extracted keywords are cataloged according to their corresponding attribute fields. The occurrence number of each attribute is counted by the actual values assumed. Assume a total number of Q attributes, $\{a_q^k \mid \forall q = 1, \dots, Q\}$, are used

to describe component $Co_{jk}^M \in \{Co_{jk}^M\}_{K_j}$. Dividing the number of occurrences by the total number of records scanned from the text file, $\{a_{qi}^k \mid \forall i = 1, \dots, P\}$, the occurrence frequency of each attribute is determined as:

$$f_q^k = \frac{c_q^k}{P}, \quad (6.4)$$

where f_q^k denotes the occurrence frequency of the q -th attribute of the k -th component, a_q^k , c_q^k is the count of active instances of a_q^k , and P is the total number of records contained in the text file, which is equal to the total number of routing instances. The occurrence frequencies explicitly suggest how often the attributes are used to characterize individual components of the same type.

(5) *Prioritize attribute fields.* If an attribute is used to describe more number of specific components, these components are more similar, in the sense that they exhibit this attribute more than others. Therefore, the relative importance of attributes in terms of occurrence frequencies should be introduced to model their relevance to the similarity measure. This coincides with the common practice that some criteria play more important roles than others in discerning similar things. The relative importance of these attributes is indicated by their weights, i.e.,

$$w_q^k = \frac{f_q^k}{\sum_{q=1}^Q f_q^k}, \quad (6.5)$$

$$\sum_{q=1}^Q w_q^k = 1, \quad (6.6)$$

where w_q^k denotes the weight of attribute a_q^k .

(6) *Determine scales for nominal values.* To compare nominal values, a semantic scale is necessary in assessing the corresponding attribute type. Usually a number between 0 and 1 is assigned for a specific nominal value, whereby 0 represents no information content and 1

indicates the maximal amount of information content. For example, the semantic scale for attribute “color” may be established by assigning 0.2, 0.3, 0.4 and 0.6 for “yellow”, “green” “red”, and “blue”, respectively. Usually such scales are determined a priori based on domain knowledge. If no domain experts are available, then simply use 1 for exactly the same nominal values and 0 for different ones, regardless of their proximity. With quantification of nominal attributes, both nominal and numerical values can be processed in the same manner, despite of their origins.

(7) *Compare attributes for their similarity.* For an attribute, $a_q^k \in \{a_q^k\}_Q$, the similarity of its instances is determined by comparing their difference (i.e., dissimilarity) in attribute values, i.e.,

$$S_{rs}^{a_q^k} = 1 - \frac{|a_{qr}^{k*} - a_{qs}^{k*}|}{\max\{a_{qi}^{k*} \mid \forall i = 1, \dots, P\} - \min\{a_{qi}^{k*} \mid \forall i = 1, \dots, P\}}, \quad (6.7)$$

where $S_{rs}^{a_q^k} \in [0, 1]$ denotes the similarity of two attribute values, $a_{qr}^{k*}, a_{qs}^{k*} \in \{a_{qi}^{k*}\}_P$.

(8) *Calculate similarity degree.* The similarity of two primitive component variants, $S_{rs}^{Co_{jk}^M}$, is calculated as a weighted sum of similarity measures of all their attributes, i.e.,

$$S_{rs}^{Co_{jk}^M} = \sum_{q=1}^Q (w_q^k S_{rs}^{a_q^k}), \quad (6.8)$$

where $0 \leq S_{rs}^{Co_{jk}^M} \leq 1$.

(9) *Construct component similarity matrices.* Steps (6)-(7) are repeated for all the instances of this component type recorded in the data file. Then a $P \times P$ matrix, $\left[S_{rs}^{Co_{jk}^M} \right]_{P \times P}$, is constructed to present pairwise similarity measures for this primitive component type. Enumerating all the primitive component files, a number of such $P \times P$ matrices, $\left\{ \left[S_{rs}^{Co_{jk}^M} \right]_{P \times P} \mid k = 1, \dots, K^{Pri} \right\}$, are constructed, in accordance with a total number of K^{Pri} primitive component types.

Similarity measure for compound components

Each l -node operation, O_j , enacts a subtree for producing a compound component, Co_j^P , from primitive components, $\{Co_{jk}^M\}_{K_j}$. Romanowski and Nagi (2005) demonstrated that, when structural differences mean less than content dissimilarity, a bottom-up approach using bipartite matching excels in finding the minimum difference between individual subtrees. Therefore, bipartite matching is applied to derive compound component similarity from the similarity measures of its primitive components.

Figure 6.3 illustrates the principle of bipartite matching. Two operation variants, $O(O')$ entail two subtrees consisting of one compound component, $A(A')$, and two primitive components, $B(B')$ and $C(C')$. First, similarity at the primitive component level is measured. Assume similarity of two child nodes B and B' is measured as $S_{BB'}=0.72$. Similarity of B and C' is $S_{BC'}=0.08$, C and B' is $S_{CB'}=0.13$, and $S_{CC'}=0.86$. The similarity at the compound component level is calculated as the average value of the maximum similarity measures of paired child nodes, that is, $S_{AA'} = \frac{S_{BB'} + S_{CC'}}{2} = \frac{0.72 + 0.86}{2} = 0.79$.

Further considering that different primitive components may contribute differently to the compound component, weighted bipartite matching (Romanowski and Nagi, 2004) can be adopted by introducing different weights to the child nodes. Thus the end result is a weighted sum, that is,

$$S_{rs}^{Co_j^P} = \sum_{k=1}^{K_j} \left(w_{jk}^M \max \left\{ S_{rs}^{(Co_{jk}^M, Co_{rg}^M)} \mid \forall k, g = 1, \dots, K_j \right\} \right), \quad (6.9)$$

where $S_{rs}^{Co_j^P}$ with $0 \leq S_{rs}^{Co_j^P} \leq 1$ gives the similarity measure of two compound components of the same type, Co_{jr}^{P*} and Co_{js}^{P*} , w_{jk}^M where $\sum_{k=1}^{K_j} w_{jk}^M = 1$) indicates the relative importance of

$\{Co_{jk}^M\}_{K_j}$ in regard to O_j , and $S_{rs}^{(Co_{jk}^M, Co_{jk}^M)}$ denotes the similarity measure of a paired child nodes, Co_{jkr}^{M*} and Co_{jgs}^{M*} . Such weight values can be determined based on domain knowledge. Compound component similarity measure using bipartite matching is further simplified because no similarity exists between components of different types. Accordingly, maximum similarity measures appear between same type components. In measuring compound component similarity, the similarity of child components (either primitive or compound types) must be obtained in advance.

Step 2: Product Similarity Measure

Each product component, $\Phi_{O_j}^P = Co_j^P$, is de facto a type of compound components. Its similarity measure follows the same text mining procedure of primitive components and bipartite matching of compound components, and thus $S_{rs}^{P_j} = S_{rs}^{Co_j^P}$. First, the data records of product components are extracted from each operation text file. Product components are described by child components rather than attributes that are used to describe primitive components. Then, based on the obtained similarity of child components in the primitive and compound component similarity matrices bipartite matching is used to measure product component similarity. As a result, a product similarity matrix, $[S_{rs}^{P_j}]_{p \times p}$, can be constructed for each product component type. Finally, enumerating all the product components files, $\{Co_j^P\}_N$, a total number of N product similarity matrices are constructed.

Step 3: Resource Similarity Measure

The resource description, R_j , of each operation, $O_j \in \{O_j\}_N$, includes three attributes: machine, M_j , cycle time, T_j , and setup, S_j . While M_j and S_j are nominal attributes, T_j is of the numerical type. Text mining is conducted in a similar fashion as that of primitive components. Resource descriptions of all operations (both l -nodes and i -nodes) are cataloged in separate text files according to operation types, i.e., resource descriptions of operations of

the same type are saved in same file. Then text mining is carried out with respect to the three attribute fields, and thus similarity measures in terms of machine ($S_{rs}^{M_j}$), cycle time ($S_{rs}^{T_j}$) and setup ($S_{rs}^{S_j}$) are derived, i.e.,

$$S_{rs}^{M_j} = I - \frac{|M_{jr}^* - M_{js}^*|}{\max\{M_{ji}^* \mid \forall i = 1, \dots, P\} - \min\{M_{ji}^* \mid \forall i = 1, \dots, P\}}, \quad (6.10)$$

$$S_{rs}^{T_j} = I - \frac{|T_{jr}^* - T_{js}^*|}{\max\{T_{ji}^* \mid \forall i = 1, \dots, P\} - \min\{T_{ji}^* \mid \forall i = 1, \dots, P\}}, \quad (6.11)$$

$$S_{rs}^{S_j} = I - \frac{|S_{jr}^* - S_{js}^*|}{\max\{S_{ji}^* \mid \forall i = 1, \dots, P\} - \min\{S_{ji}^* \mid \forall i = 1, \dots, P\}}, \quad (6.12)$$

where $S_{rs}^{w_j}, S_{rs}^{T_j}, S_{rs}^{S_j} \in [0, 1]$, M_{ji}^* , T_{ji}^* and S_{ji}^* stand for the specific values of machine, cycle time and setup of operation O_j , respectively,

Accordingly, the resource similarity measure of two operation variants, $S_{rs}^{R_j}$, is calculated as a weighted sum of similarity measures regarding all their attributes, i.e.,

$$S_{rs}^{R_j} = w^{M_j} S_{rs}^{M_j} + w^{T_j} S_{rs}^{T_j} + w^{S_j} S_{rs}^{S_j}, \quad (6.13)$$

where $0 \leq S_{rs}^{R_j} \leq 1$, $w^{M_j} + w^{T_j} + w^{S_j} = 1$, and w^{M_j} , w^{T_j} and w^{S_j} denote the relative importance (or weight) of machine, cycle time and setup attributes in regard to operation O_j , respectively.

The weights of machine, cycle time and setup can be determined based on domain knowledge.

Enumerating all instances of resource description, R_j , a resource similarity matrix, $[S_{rs}^{R_j}]_{P \times P}$, is constructed to present pairwise resource comparisons of all variants of operation O_j .

Similarly, a total number of N resource similarity matrices are constructed for all the operations, $\{O_{jk}^*\}_{N \times P}$.

Step 4: Node Content Similarity Measure

With the availability of material similarity, product similarity, and resource similarity, the operations of the same types are compared using Eq. (6.2). A $P \times P$ operation matrix,

$[S_{rs}^{O_j}]_{P \times P}$, is established to document the similarity of the same type operations. Enumerating all operation types, a total number of $N \ P \times P$ operation similarity matrices are constructed.

Based on the above N number operation similarity matrices, the similarity of two routings can be computed according to Eq. (6.1). Enumerating P routings, their pairwise node content similarity measures are obtained.

Step 5: Node Content Similarity Normalization

Since $0 \leq S_{rs}^{M_j}, S_{rs}^{P_j}, S_{rs}^{R_j} \leq 1$, $S_{rs}^{O_j}$ and S_{rs}^{NC} may not suggest a relative measure ranging from 0 to 1. They need to be normalized to achieve a consistent comparison. Many normalization methods are available such as the z-score and the max-min normalization methods (Han and Kamber, 2001). This research adopts the most commonly used one: max-min method to convert the node content similarity measure to a relative magnitude between 0 and 1, as follows:

$$S_{rs}^{NC'} = \frac{S_{rs}^{NC} - \min\{S_{rs}^{NC} | \forall r, s = 1, \dots, P\}}{\max\{S_{rs}^{NC} | \forall r, s = 1, \dots, P\} - \min\{S_{rs}^{NC} | \forall r, s = 1, \dots, P\}}, \quad (6.14)$$

where S_{rs}^{NC} and $S_{rs}^{NC'}$ denotes the original and normalized node content similarity measures between R_r and R_s , respectively. As a result, a node content similarity matrix, $[S_{rs}^{NC'}]_{P \times P}$, is constructed. Each matrix element indicates the normalized node content similarity of two routing variants corresponding to row and column, respectively.

6.3.1.2. Tree Structure Similarity

Tree structure similarity measures the degree of commonality between two routings in terms of their operations sequences (i.e., the arcs of precedence graphs). To deal with such structural data, the tree matching technique is applied, which acquires the difference between two trees by finding the likeness of their structures (Valiente, 2002). The procedure proceeds as follows.

(1) *Determine a base R.* For a specific routing precedence graph, each particular operation executes in a different sequence from one another. As a result, a routing constitutes a partial order set, in that not all the items in the set follow the same binary relation (<http://www.nist.gov/dads/HTML/partialorder.html>). Since a partial order can be represented by more than one tree (Martinez et al., 2000), each routing may possess a number of alternative representation trees. The similarity measure of two routings may vary if different representation trees of them are used for the comparison. It is thus necessary to make decisions based on pairwise comparisons of all possible representation trees of two routings.

Due to the symmetric property of distance measure and cyclic representation of a partial order (Martinez et al., 2000), the pairwise comparisons can be simplified to merely compare an arbitrary tree of one R (referred to as “base R”) with all representation trees of each of the other routings. To reduce the total number of pairwise comparisons between any two routings, the one with the most representation trees should be selected as the base R. The number of representation trees of a routing is given as 2^N , where N is the number of nodes with two child nodes in a tree (Martinet et al., 2000).

(2) *Generate representation trees.* For a number of P given routings ordered according to the descending number of their representation trees, each of the first $(P-1)$ routings $\{R_r \mid \forall r = 1, \dots, P-1\}$, serves as a base R for comparison of tree structure similarity with its following Rs, $\{R_{r+1} \mid \forall r = 1, \dots, P-1\}$. In other words, one routing only needs to be compared with subsequent routings. Thus a total number of $P \times (P-1)/2$ pairwise comparisons are needed. Except for the routing selected to be the first base R, e.g., R_1 , all the remaining Rs, e.g., $\{R_r \mid \forall r = 2, \dots, P\}$, are compared with their corresponding base Rs, e.g., $\{R_{r-1} \mid \forall r = 2, \dots, P\}$. The reason is that no routing possesses nodes with two children more than R_1 . Consequently, all corresponding representation trees need to be generated for $R_r \mid \forall r = 2, \dots, P$.

(3) *Establish a tree edit graph.* The basic principle of tree matching is to compare two trees based on tree transformation – to transform one tree to exactly the same as the other one (Valiente, 2002). It is accomplished by adopting a series of tree editing operations, including node insertion, deletion, and substitution. The transformation manifests certain costs incurred by the editing operations. The value of editing costs indicates the distance between two trees, namely their dissimilarity.

Since no constraints are introduced to the order of executing these editing operations, a number of ways may exist to convert one tree to another. It would, however, be difficult to enumerate all possible ways of transformation in order to compare two trees. As a consequence, the smallest transformation cost obtained from enumeration may not be the optimal measure to suggest the distance of two trees. To circumvent this problem, the tree edit graph (Valiente, 2002) is employed in this research to provide an indirect way of tree transformation.

Figure 6.5 shows a tree edit graph for transformation between two trees, T_1 and T_2 . Each vertex indicated by a black dot represents a combination of two paired nodes from two trees, for example, $V_i W_j$: V_i from T_1 and W_j from T_2 . The arcs between two vertices represent certain tree editing operations. A horizontal arc between two adjacent vertices, for example, $(V_i W_j, V_i W_{j+1})$, means insertion of a node into the transformed tree, that is, to insert W_{j+1} into tree T_2 . A vertical arc, for example, $(V_i W_j, V_{i+1} W_j)$, indicates the deletion of a node from the transforming tree; that is, to delete node V_i from tree T_1 . A diagonal arc, for example, $(V_i W_j, V_{i+1} W_{j+1})$, denotes the substitution of one node in the transformed tree with another node in the transforming tree, that is, to substitute W_{j+1} in T_2 with V_{i+1} in T_1 . However, not all arcs in the graph are valid editing operations for tree transformation. The validity of arcs is subject to the these rules: (1) for a valid horizontal arc $(V_i W_j, V_i W_{j+1})$, $D_{V_{i+1}} < D_{V_{j+1}}$; (2) for a valid

vertical arc $(V_i W_j, V_{(i+1)} W_j)$, $D_{V_{(i+1)}} > D_{V_{(j+1)}}$; and (3) for a valid diagonal arc $(V_i W_j, V_{(i+1)} W_{(j+1)})$, $D_{V_{(i+1)}} = D_{V_{(j+1)}}$, where D_x stands for the depth of node x .

The cost of an editing operation is reflected as a value attached to the corresponding arc. To facilitate comparisons based on a consistent common ground, the unit cost values for different types of tree editing operations are assumed to be the same. Therefore, the cost of transforming one tree to another is indicated by the number of involved editing operations per se. For tree similarity measure between a routing with its base routing, the total number of tree edit graphs need to be generated is equal to the number of representation trees of the routing rather than its base routing.

Figure 6.5 An example of tree edit graph

(4) *Find the shortest path for the distance measure.* In a tree edit graph, there are many paths from the top-left corner to the bottom-right corner. Each path suggests a possible way of transforming one tree to another, which carries different costs as well. The distance between two trees is measured according to the shortest path among all the transformation ways. Such shortest path has a minimum number of arcs, i.e., the fewest editing operations, and thus the minimum cost, which is defined as the distance between two representation trees. The distance measure, ${}^{RT}D_{rs}$, between each pair of representation trees is hence given as:

$${}^{RT}D_{rs} = A^* C, \quad (6.15)$$

where A^* is the total number of valid arcs in the shortest path, and C is a constant indicating unit cost value associated with each operation, regardless of its type.

This procedure is repeated for comparing all the representation tree pairs for one routing with its base routing. The distance measure, D_{rs}^{TS} , between the two routings is determined by the minimum distance among all distance measures between its representation trees and the arbitrary one of its base R, i.e., $D_{rs}^{TS} = \min\{D_{rsi}^{TS} | \forall i = 1, \dots, X\}$, wherein X is the number of representation trees of the non-base routing. This procedure is repeated for all $P \times (P-1)/2$ routing pairs, and the pairwise distance measures of all P routings are obtained.

(5) *Normalize distance data.* The above distance measures are all absolute values instead of relative magnitude. For consistent comparison, they need to be normalized. The max-min method is adopted to convert the absolute distance measure of each routing pair to a dimensionless value ranging between 0 and 1, as follows:

$$D_{rs}^{TS'} = \frac{D_{rs}^{TS} - \min\{D_{rs}^{TS} | \forall r, s = 1, \dots, P\}}{\max\{D_{rs}^{TS} | \forall r, s = 1, \dots, P\} - \min\{D_{rs}^{TS} | \forall r, s = 1, \dots, P\}}, \quad (6.16)$$

where D_{rs}^{TS} and $D_{rs}^{TS'}$ denotes the absolute and normalized distance measures between R_r and R_s , respectively.

(6) *Calculate tree structure similarity.* According to the normalized distance measure, the similarity can be calculated as follows:

$$S_{rs}^{TS} = 1 - D_{rs}^{TS'}. \quad (6.17)$$

(7) *Construct a routing structure similarity matrix.* Calculate similarity values for all the routings in the routing data set. Then present all pairwise similarity measures in a $P \times P$ matrix, $[S_{rs}^{TS}]_{P \times P}$. Each matrix element indicates the structure similarity of two routings corresponding to row and column, respectively.

6.3.1.3. Routing Similarity

As node content similarity, $S_{rs}^{NC'} \in [0, 1]$, and tree structure similarity, $S_{rs}^{TS} \in [0, 1]$, are two independent measures, the overall routing similarity, S_{rs} , is composed by an Euclidian distance, i.e.,

$$S_{rs} = \sqrt{(S_{rs}^{NC'})^2 + (S_{rs}^{TS})^2}. \quad (6.18)$$

Routing similarity calculation is repeated for all routings in the given data set. The pairwise node content similarity measures are obtained. To convert S_{rs} to a consistent magnitude for comparison ranging from 0 to 1, the normalization process is applied. The normalized routing similarity value, S'_{rs} , is calculated according to the following equation:

$$S'_{rs} = \frac{S_{rs} - \min\{S_{rs} | \forall r, s = 1, \dots, P\}}{\max\{S_{rs} | \forall r, s = 1, \dots, P\} - \min\{S_{rs} | \forall r, s = 1, \dots, P\}}, \quad (6.19)$$

such that $0 \leq S'_{rs} \leq 1$.

All pairwise routing similarity measures are presented in a $P \times P$ matrix, $[S'_{rs}]_{P \times P}$. Each matrix element indicates the normalized similarity measure of two routings corresponding to row and column, respectively. The logic of routing similarity measures is summarized in Figure 6.6.

6.3.2. Routing Clustering

Routing clustering aims to group a set of individual routings into classes of similar routings. A routing cluster is a collection of routings that are similar to one another within the same cluster yet dissimilar to the routings in other clusters (Han and Kamber, 2001). Considering the complex data types involved in routings, this research adopts a fuzzy clustering approach. Fuzzy clustering excels in revealing the similarity between any two objects involving subjectiveness and imprecision (Zimmermann, 2001).

Figure 6.6 Logic of routing similarity measure

6.3.2.1. Principle of Fuzzy Clustering

Fuzzy clustering seeks to create a hierarchical decomposition of a given set of objects, in which each object forms a separate group and successively the objects or groups close to one another are merged at different similarity levels. In this research, historical data about individual routing contained in the data files can be used to measure the similarity degree based on the compatibility of routing data. In comparison with the k-means method, fuzzy clustering partitions routing instances based on the similarity degree that is derived from the real data of operations, rather than based on subjectively pre-defined clusters.

Given a collection of objects, e.g., routings, $Z = \Omega = \{R_i \mid \forall i = 1, \dots, P\}$, a fuzzy set $\tilde{F}$ in Z is defined as a set of ordered pairs: $\tilde{F} = \{(z, \varphi_{\tilde{F}}(z)) \mid z \in Z\}$, where $\varphi_{\tilde{F}}(z)$ is called the membership function of z in $\tilde{F}$ that maps Z to the membership space $[0, 1]$. The membership function is also referred to as the degree of compatibility or degree of truth. A certain set of objects that belongs to the fuzzy set $\tilde{F}$ at least to the degree λ is called the λ -cut set or simply the λ -cut.

Assume Z is a finite, non-empty universal set. Let $\tilde{R}$ be a fuzzy relation, i.e., a fuzzy subset, in $Z \times Z$, that is, $\tilde{R} = \{(x, y), \varphi_{\tilde{R}}(x, y) \mid \forall (x, y) \in Z \times Z\}$.

- (1) $\tilde{R}$ is called reflexive if $\varphi_{\tilde{R}}(x, x) = 1 \mid \forall x \in Z$;
- (2) $\tilde{R}$ is called symmetric if $\varphi_{\tilde{R}}(x, y) = \varphi_{\tilde{R}}(y, x) \mid \forall x, y \in Z$,
- (3) $\tilde{R}$ is called max-min transitive if $\tilde{R}(x, z) = \max_y [\min\{\tilde{R}(x, y), \tilde{R}(y, z)\}] \mid \forall x, y, z \in Z$,

i.e., $\tilde{R} \circ \tilde{R} \subseteq \tilde{R}$, where $\circ$ is max-min composition (Zimmermann, 2001);

- (4) $\tilde{R}$ is a similarity relation if $\varphi_{\tilde{R}}(x, x) = 1 \mid \forall x \in Z$ and $\varphi_{\tilde{R}}(x, y) = \varphi_{\tilde{R}}(y, x) \mid \forall x, y \in Z$;

and

(5) $\tilde{R}$ is an equivalence relation if $\varphi_{\tilde{R}}(x, x) = 1 \mid \forall x \in Z$, $\varphi_{\tilde{R}}(x, y) = \varphi_{\tilde{R}}(y, x) \mid \forall x, y \in Z$, and $\tilde{R}(x, z) = \max_y \left[\min \left\{ \tilde{R}(x, y), \tilde{R}(y, z) \right\} \right] \mid \forall x, y, z \in Z$ (Yang and Gao, 1996).

When Z is a finite set, the similarity relation $\tilde{R}$ is a similarity matrix (Yang and Gao, 1996). Hence fuzzy clustering becomes a set of N objects of Z to be clustered, given a fuzzy similarity matrix $\tilde{R}$ defined on Z . Assume $\tilde{R}^n$ denotes the n -th power of fuzzy similarity matrix $\tilde{R}$, i.e., $\tilde{R}^n = \tilde{R}^{n-1} \circ \tilde{R}$. Then the minimal max-min transitive closure of $\tilde{R}$, denoted as $\hat{R}$, is defined as $\hat{R} = \bigcup_{i=1}^N \tilde{R}^i = \tilde{R}^N$. Therefore, $\hat{R}$ is a fuzzy equivalence matrix.

Assume $0 \leq \lambda \leq 1$ and let $\hat{R}_\lambda = \{(x, z) \mid \varphi_{\hat{R}}(x, z) \geq \lambda, \forall x, z \in Z\}$. Then, according to Wang and McCauley-Bell (1996):

(1) $\hat{R}_\lambda$ is an equivalence matrix on Z ; and

(2) Let $G_{\hat{R}_\lambda}$ denote the partition on Z induced according to $\hat{R}_\lambda$. Then for each $B \in G_{\hat{R}_\lambda}$, there exists $E \in G_{\hat{R}_{\lambda'}}$, so that $B \subseteq E$, as long as $\lambda' \leq \lambda$.

As λ is increased, a finer partition can be achieved. With a hierarchy of partitions of objects, k -clusters of objects can be identified. Figure 6.7 simply illustrates the fuzzy clustering with respect to a fuzzy equivalence matrix defined on five objects. Given different values of similarity threshold, λ , different clustering results can be obtained.

$$\hat{R} = \begin{matrix} & \begin{matrix} O_1 & O_2 & O_3 & O_4 & O_5 \end{matrix} \\ \begin{matrix} O_1 \\ O_2 \\ O_3 \\ O_4 \\ O_5 \end{matrix} & \begin{bmatrix} 1 & 0.4 & 0.8 & 0.5 & 0.5 \\ 0.4 & 1 & 0.4 & 0.4 & 0.4 \\ 0.8 & 0.4 & 1 & 0.5 & 0.5 \\ 0.5 & 0.4 & 0.5 & 1 & 0.6 \\ 0.5 & 0.4 & 0.5 & 0.6 & 1 \end{bmatrix} \end{matrix}$$

A fuzzy equivalence matrix $\hat{R}$ defined on O

Nested partitions of O according to $\hat{R}_\lambda$

Different clusters resulted from different λ values

Figure 6.7 Illustration of fuzzy clustering

6.3.2.2. Procedure of Routing Fuzzy Clustering

(1) *Define a fuzzy similarity matrix.* A fuzzy similarity matrix, sR , is defined as similarity measures for a given set of routings, $\Omega = \{R_1, \dots, R_p\}$. The sR is constructed in a matrix form, that is, ${}^sR = [S'_{rs}]_{p \times p}$, where $0 \leq S'_{rs} \leq 1$ suggests a pairwise relationship (similarity grade) between any two routing instances. Within the context of routing clustering, sR is identical to the routing similarity matrix obtained through similarity measure.

In a routing similarity matrix, it holds true that $S'_{rr} = 1 \mid \forall r = 1, \dots, P$, suggesting that sR is reflexive. Also true is $S'_{rs} = S'_{sr}$, $\forall r, s = 1, \dots, P$, suggesting that sR is symmetrical. Therefore, matrix ${}^sR = [S'_{rs}]_{p \times p}$, $S'_{rs} \in [0, 1]$ becomes a fuzzy similarity matrix defined on Ω .

(2) *Construct a fuzzy equivalence matrix.* A fuzzy equivalence matrix is defined for Ω with transitive closure of a fuzzy similarity matrix (Zimmermann 2001). The fuzzy similarity matrix sR is a fuzzy equivalence matrix if and only if the transitive condition can be met, i.e.,

$$S'_{rs} \geq \max\{\min\{S'_{rs}, S'_{zs}\} \mid \forall R_r, R_z, R_s \in \Omega\}. \quad (6.20)$$

To convert a fuzzy similarity matrix sR to a fuzzy equivalence matrix ${}^e\hat{R}$, the “continuous multiplication” method is adopted. Multiplication in fuzzy matrixes is also known as max-min composition (Lin and Lee, 1996). The underlying principle of max-min composition is as follows.

$${}^e\hat{R} = {}^sR^{2k} = ({}^sR \circ {}^sR)^k, \quad (6.21)$$

(3) *Determine a λ -cut of the equivalence matrix.* The λ -cut is a crisp set, ${}^sR_\lambda$, of Ω , such that the similarity grade of sR is no less than λ , that is,

$${}^sR_\lambda = [\tau_{rs}]_{P \times P}, \quad (6.22)$$

where

$$\tau_{rs} = \begin{cases} 1 & \text{if } S'_{rs} \geq \lambda \\ 0 & \text{if } S'_{rs} < \lambda \end{cases}, \quad S'_{rs} \in [0, 1]. \quad (6.23)$$

Then each λ -cut, ${}^sR_\lambda$, is an equivalence matrix representing the presence of similarity among routing instances to the degree λ .

(4) *Identify routing clusters.* A netting graph method (Yang and Gao, 1996) is applied to identify partitions of routing instances with respect to a given equivalence matrix. The procedure of generating a fuzzy netting graph is summarized as the following:

- (a) *Replace all 1s in the λ -cut matrix with *;*
- (b) *Remove all 0s;*
- (c) *Put the routing name, i.e., R_r , in the corresponding diagonal cell;*
- (d) *Draw both vertical and horizontal lines between each * and the corresponding two routings, i.e., R_r and R_s , in the diagonal; and*
- (e) *Group routings that are connected by these lines into same clusters.*

The value of $\lambda \in [0, 1]$ indicates the similarity threshold of a λ -cut. Given an equivalence matrix, different clustering results may be obtained according to individual similarity thresholds. In practice, the value of λ is often determined by domain experts with

many practical considerations, such as the extent of process reuse, convenience of process configuration, and capabilities of the product and process platforms.

6.3.3. Routing Unification

Routing unification attempts to unify all members of each routing cluster into a generic routing of this cluster. As shown in Figure 6.8, the major elements of a generic routing, $GR = \langle A, \succ \rangle$, include a set of master routings and a set of selective routings. The GR entails the configuration mechanism underlying a process family per se.

The master routing set comprises sets of master operation classes, A^M , and master precedence classes, $\succ^M$, which constitute the common building blocks of a process family (i.e., the corresponding routing cluster). Within this process family, all individual routings conform to a common routing structure while exhibiting variations in specific operations characteristics.

Selective routings are sets of selective operation classes, A^S , and selective precedence classes, $\succ^S$, which enable differentiation of individual routings. One routing variant may possess a different routing structure from one another by adding or deleting different selective routings to the master routings in order to meet specific production requirements. The inclusion of an operation in a routing is indicated by the presence of its related precedence. Each precedence, $O_i \succ O_j \mid \forall i \neq j \in [1, N]$, is a binary variable with a 0/1 value, suggesting the existence of operation sequence $O_i \succ O_j$, and in turn the inclusion of operation O_i .

Figure 6.8 Principle of a generic routing

While all master operations must be assumed by any routing variant, namely $O_i^M \succ O_j^M \equiv 1$ for all $O_i^M, O_j^M \in A^M$ and $O_i^M \succ O_j^M \in \succ^M$, their characteristics in components and resources may be different among individual routings. This is modeled by specifying *execution rules* for the precedence class concerned. Execution rules are logical expressions defined for the instantiation of master operation classes by clarifying the circumstances under which individual operations are carried out. The general form of execution rules is like a production rule: “IF antecedent THEN consequent”. Logic operations such as AND, OR and XOR can be applied to both the antecedent and the consequent. Usually the antecedent is defined in terms of combinations of specific instances of operations material or product components, whilst the consequent depicts particular characteristics of related operations, for example, “IF Co^{M*}_{311} OR Co^{M*}_{321} AND Co^{P*}_{21} THEN R^*_{31} AND M^*_{21} AND S^*_{11} AND T^*_{11} ”.

A selective operation, $O_i^S \in A^S$, is adopted if and only if its related precedence assumes a valid value, i.e., $\exists O_i^S \succ O_j^S = 1$, where $O_i^S \succ O_j^S \in \succ^S$. When $\forall O_i^S \succ O_j^S = 0$, the selective routing (both O_i^S and O_j^S) by no means presents in a routing variant. The instantiation

of selective routings is modeled by specifying *inclusion conditions*. An inclusion condition is expressed as production rules as well, where the antecedent bears the same implications as that of execution rules, and the consequent however is defined by the existence of the related selective precedence, for example, “IF Co_{31l}^{M*} AND Co_{32l}^{M*} THEN $O_{6l}^* \succ O_{3l}^* = 0$ AND $O_{7l}^* \succ O_{3l}^* = 0$ ”.

Built upon the master and selective routing elements, the generic routing is formed by maintaining a valid tree structure. The generic tree structure, $\mathbf{G}$, is developed, through a tree growing process, from the general tree structures embedded in individual routings, referred to as basic trees, $\{\mathbf{T}_z\}_Z$, within a routing cluster. The formation of a GR involves four major steps, including assorting basic routing elements, identifying master and selective routing elements, forming basic trees, and tree growing, as discussed next.

Step 1: Basic Routing Element Assortment. The first step of routing unification is to breakdown individual routings into operations and precedence elements. For each member of a routing cluster, $R_r \in \{R_r \mid \forall r = 1, \dots, M \leq P\}$, the nodes (operations) and arcs (precedence) of the corresponding routing tree are assorted and categorized by *l*-nodes or *i*-nodes. This results in a *l*-node set, $\{LN_{jr}^*\}_{N^{LN} \times M}$, an *i*-node set, $\{IN_{jr}^*\}_{N^{IN} \times M}$, a *l*-node arc set, $\{LA_{jr}^*\}_{N^{LN} \times M}$, and an *i*-node arc set, $\{IA_{jr}^*\}_{(N^{IN} - 1) \times M}$, corresponding to *l*-node type $\{LN_j\}_{N^{LN}}$, *i*-node type $\{IN_j\}_{N^{IN}}$, *l*-node arc type $\{LA_j\}_{N^{LN}}$, and *i*-node arc type $\{IA_j\}_{N^{IN} - 1}$, respectively, where $N^{LN} + N^{IN} = N$, $\forall LN_j, IN_j \in A$, $\{LN_j\} \cap \{IN_j\} = \emptyset$, $\forall LA_j, IA_j \in \succ$, and $\{LA_j\} \cap \{IA_j\} = \emptyset$.

Configuration rules are also compiled for every node and arc contained in the routing cluster. For exactly the same operations in different routings, their related rules need to be documented only once. All rules are initially placed in one rule set.

Step 2: Master and Selective Routing Element Identification. The second step is to generalize each individual routing element (operation or precedence variant) with regard to

its original type. This is achieved by replacing the specific name or ID of each specific node or arc with the general name or ID of the operation or precedence class to which it belongs. As a result, each particular routing element is labeled with its class identification. And in turn each operation or precedence class assumes a certain number of occurrences in terms of the number of times individual routing elements are generalized into this class. Such an occurrence count performs as a commonality index revealing to what extent each routing element is reused among individual members of a routing cluster.

Given a routing cluster, $\{R_r\}_M$, if the occurrence count of a precedence class ($O_i \succ O_j$) reaches the maximal number of instances of this class contained in the cluster, i.e., $\varphi_{O_i \succ O_j} = M$, it means that all individual routings in the cluster employ this precedence class. Therefore, this precedence class along with the related operation classes suggest themselves to be the master routing elements, i.e., the master precedence and operation classes, respectively. Should $1 \leq \varphi_{O_i \succ O_j} < M$, the related operation and precedence classes are defined as selective operation and precedence classes, respectively, as not all individual variants assume them. In this way, all basic routing elements are grouped into either master or selective routing elements, that is,

$$\{LN_j\}_{N^{LN}} = \{LN_j^M \mid \forall LN_j^M \in \mathcal{A}^M\}_{N^{LNM}} \cup \{LN_j^S \mid \forall LN_j^S \in \mathcal{A}^S\}_{N^{LNS}}, \quad (6.24a)$$

$$\{IN_j\}_{N^{IN}} = \{IN_j^M \mid \forall IN_j^M \in \mathcal{A}^M\}_{N^{INM}} \cup \{IN_j^S \mid \forall IN_j^S \in \mathcal{A}^S\}_{N^{INS}}, \quad (6.24b)$$

$$\{LA_j\}_{N^{LN}} = \{LA_j^M \mid \forall LA_j^M \in \succ^M\}_{N^{LNM}} \cup \{LA_j^S \mid \forall LA_j^S \in \succ^S\}_{N^{LNS}}, \quad (6.24c)$$

$$\{IA_j\}_{N^{IN-1}} = \{IA_j^M \mid \forall IA_j^M \in \succ^M\}_{N^{INM-1}} \cup \{IA_j^S \mid \forall IA_j^S \in \succ^S\}_{N^{INS-1}}, \quad (6.24d)$$

where $N^{LNM} + N^{LNS} = N^{LN}$ and $N^{INM} + N^{INS} = N^{IN}$.

In accordance with the identified master and selective routing elements, the configuration rules are orchestrated and classified into one execution rule set for master routings and one inclusion condition rule set for selective routings. The purpose of grouping all execution rules or inclusion conditions into one separate rule set is to ensure that the

execution rules or inclusion conditions are defined globally, rather than associated with individual master or selective routing elements. This helps maintain the respective configuration rules more concisely and in consistent forms.

Step 3: Basic Tree Structure Formation. The third step deals with the generalization of basic trees underlying all the members in the routing cluster. A basic tree refers to the common tree structure assumed by some individual routings. Given the set of routing variants in a routing cluster, some may possess exactly the same tree structure but differ from one another in their operation characteristics. Therefore, a number of $Z \leq M$ basic tree structures, $\{\mathcal{T}_z\}_Z$, are identified from M member trees of the cluster.

While each member tree denotes a specific routing variant, a basic tree represents a class of individual routing variants bearing the same tree structure. To track commonality of a basic tree with respect to its represented routing variants, each arc of the basic tree is assigned a weight M_z , i.e., the number of represented routings assuming the basic tree structure, indicating its repetition degree. Initially, this weight value is assigned to all arcs of a basic tree, regardless of whether it is a master or selective precedence.

In accordance with the assortment of basic routing elements, a basic tree is specified by a 4-tuple, denoted as:

$$\mathcal{T}_z = (\mathcal{L}^N, \mathcal{I}^N, \mathcal{L}^A, \mathcal{I}^A), \quad (6.25)$$

where $\mathcal{L}^N(\mathcal{T}_z) = \{Ln_{zj}\}_{Z \times N_z^{LN}} \subseteq \{LN_j\}_{N^{LN}}$, $\mathcal{I}^N(\mathcal{T}_z) = \{In_{zj}\}_{Z \times N_z^{IN}} \subseteq \{IN_j\}_{N^{IN}}$, $\mathcal{L}^A(\mathcal{T}_z) = \{La_{zj}\}_{Z \times N_z^{LA}} \subseteq \{LA_j\}_{N^{LA}}$ and $\mathcal{I}^A(\mathcal{T}_z) = \{Ia_{zj}\}_{Z \times (N_z^{IN} - 1)} \subseteq \{IA_j\}_{N^{IN} - 1}$ are sets of basic routing elements, encompassing all l -node classes, i -node classes, l -node arc classes and i -node arc classes contained in $\mathcal{T}_z$, respectively. Their respective variant sets are $\{Ln_{zjr}^*\}_{Z \times N_z^{LN} \times M_z}$,

$\{In_{zjr}^*\}_{Z \times N_z^{IN} \times M_z}$, $\{La_{zjr}^*\}_{Z \times N_z^{LA} \times M_z}$ and $\{Ia_{zjr}^*\}_{Z \times (N_z^{IN} - 1) \times M_z}$, where $\sum_{z=1}^Z (N_z^{LN} + N_z^{IN}) \leq N$.

The basic routing elements constituting basic trees are further classified into master or selective sets based on the occurrence counts of individual precedence variants $\varphi_{o_i^z \succ o_j^z} = Z$

or $I \leq \varphi_{o_i^z \succ o_j^z} Z$, that is,

$$\{Ln_{zj}\}_{Z \times N_z^{LN}} = \{Ln_{zj}^M \mid \forall Ln_{zj}^M \in \{LN_j^M\}_{N_z^{LNM}}\}_{N_z^{LNM}} \cup \{Ln_{zj}^S \mid \forall Ln_{zj}^S \in \{LN_j^S\}_{N_z^{LNS}}\}_{N_z^{LNS}}, \quad (6.26a)$$

$$\{In_{zj}\}_{Z \times N_z^{IN}} = \{In_{zj}^M \mid \forall In_{zj}^M \in \{IN_j^M\}_{N_z^{INM}}\}_{N_z^{INM}} \cup \{In_{zj}^S \mid \forall In_{zj}^S \in \{IN_j^S\}_{N_z^{INS}}\}_{N_z^{INS}}, \quad (6.26b)$$

$$\{La_{zj}\}_{Z \times N_z^{LN}} = \{La_{zj}^M \mid \forall La_{zj}^M \in \{LA_j^M\}_{N_z^{LNM}}\}_{N_z^{LNM}} \cup \{La_{zj}^S \mid \forall La_{zj}^S \in \{LA_j^S\}_{N_z^{LNS}}\}_{N_z^{LNS}}, \quad (6.26c)$$

$$\{Ia_{zj}\}_{Z \times (N_z^{IN} - 1)} = \{Ia_{zj}^M \mid \forall Ia_{zj}^M \in \{IA_j^M\}_{N_z^{INM} - 1}\}_{N_z^{INM} - 1} \cup \{Ia_{zj}^S \mid \forall Ia_{zj}^S \in \{IA_j^S\}_{N_z^{INS} - 1}\}_{N_z^{INS} - 1}, \quad (6.26d)$$

where $N_z^{LNM} + N_z^{LNS} = N_z^{LN}$ and $N_z^{INM} + N_z^{INS} = N_z^{IN}$.

Step 4: Tree Growing. The fourth step aims to form the generic tree by pasting all basic trees one by one, that is,

$$\mathbf{G} = \mathbf{T}_1 \cup \mathbf{T}_2 \cdots \cup \mathbf{T}_z, \quad (6.27)$$

where $L^N(\mathbf{G}) = L^N(\mathbf{T}_1) \cup L^N(\mathbf{T}_2) \cdots \cup L^N(\mathbf{T}_z)$, $I^N(\mathbf{G}) = I^N(\mathbf{T}_1) \cup I^N(\mathbf{T}_2) \cdots \cup I^N(\mathbf{T}_z)$, $L^A(\mathbf{G}) = L^A(\mathbf{T}_1) \cup L^A(\mathbf{T}_2) \cdots \cup L^A(\mathbf{T}_z)$, and $I^A(\mathbf{G}) = I^A(\mathbf{T}_1) \cup I^A(\mathbf{T}_2) \cdots \cup I^A(\mathbf{T}_z)$.

Tree growing starts with the selection of a seed, i.e., an initial generic tree, $\mathbf{G}_1$. Among basic trees, $\{\mathbf{T}_z\}_Z$, the one holding a longest path and possessing the maximal number of i -nodes is recognized as the seed. Such a comprehensive tree encompasses most production conditions occurring among the process family members. Then the initial generic tree, $\mathbf{G}_1$, starts to grow by unifying, one by one, with the other $Z - 1$ basic trees, that is,

$$\mathbf{G}_i = \mathbf{G}_{i-1} \cup \mathbf{T}_i, \quad (6.28)$$

where $\mathbf{G}_i$ is the tree after unifying the i -th basic tree, $\mathbf{T}_i$, with the previous $(i-1)$ -th unifying result $\mathbf{G}_{i-1}$. After all basic trees are unified, the growing tree reaches its final form, $\mathbf{G}_Z$, i.e., $\mathbf{G}$.

Since the GR includes all operations occurred in the routing cluster, both $L^N(\mathbf{G})$ and $I^N(\mathbf{G})$ are simply a union of all node sets contained in basic trees, i.e.,

$$L^N(\mathbf{G}_i) = L^N(\mathbf{G}_{i-1}) \cup \{Ln_{ij}\}_{I \times N_i^{LN}}, \quad (6.29a)$$

$$I^N(\mathbf{G}_i) = I^N(\mathbf{G}_{i-1}) \cup \{In_{ij}\}_{I \times N_i^{IN}}. \quad (6.29b)$$

However, $L^A(\mathbf{G})$ and $I^A(\mathbf{G})$ do not work with simple union operations, because a tree structure has to be maintained throughout the tree growing process. Since all the master l -node arcs and master i -node arcs are contained in both $\mathbf{T}_i$ and in $\mathbf{G}_{i-1}$, the weights of these master arcs are incremented in $\mathbf{G}_{i-1}$ by the weight values of the same arcs in $\mathbf{T}_i$. The selective l -node arcs and selective i -node arcs of $\mathbf{T}_i$, however, may not always contribute to maintaining the generic structure. If adding some arcs of $\mathbf{T}_i$ to the growing tree jeopardizes the generic tree structure, then these arcs are put in an additional arc set, A_{AS} , for further examination after increasing the weights by the value of the same arcs (already in A_{AS}), which result from previous tree unification.

For any l -node arc that exists in $\mathbf{T}_i$ but not in $\mathbf{G}_{i-1}$, it is of selective type, i.e., La_{ij}^S . Such selective arcs, $\{La_{ij}^S\}_{N_i^{LNS}}$, are copied to $\mathbf{G}_{i-1}$ only when their associated operations, i.e., selective l -nodes, $\{Ln_{ij}^S\}_{N_i^{LNS}}$, do not exist in $\mathbf{G}_{i-1}$. In other situations, inclusion of a selective l -node arc of $\mathbf{T}_i$, $La_{ij}^S \in L^A(\mathbf{T}_i)$, into $\mathbf{G}_{i-1}$ or putting it in A_{AS} depends on the result of comparing its weight, $W_j^{La_{ij}^S}$, with that, $W_j^{La_{(i-1)j}^S}$, of the corresponding arc in $\mathbf{G}_{i-1}$. The arc assuming higher weight value should be included in $\mathbf{G}_{i-1}$, as a higher weight implies more common. Such a weight results from the sum of the occurrence count of this arc in all member trees, i.e., M_z , and the recorded weight of the same arc in A_{AS} , if it can be found in A_{AS} . Likewise, for a selective i -node arc, $Ia_{ij}^S \in L^A(\mathbf{T}_i)$, does not exist in $\mathbf{G}_{i-1}$, only when

neither the associated parent i -node, $Pln_{ij}^S \in I^N(\mathcal{T}_i)$, nor child i -nodes, $CIn_{ij}^S \in I^N(\mathcal{T}_i)$, exists in $\mathcal{G}_{i-1}$, can the i -node arc be added directly to $\mathcal{G}_{i-1}$. Otherwise, weight evaluation is needed.

Arc unification essentially aims to combine the arc sets of $\mathcal{T}_i$ and $\mathcal{G}_{i-1}$ while removing less common arcs. As a result, the union operations of l -node and i -node arcs are given as follows:

$$\begin{aligned} L^A(\mathcal{G}_i) = & \left\{ La_{zj}^M \mid \forall La_{zj}^M \in \succ^M \right\}_{N_z^{LNM}} \cup \left\{ La_{ij}^S \mid \forall La_{ij}^S \in \succ^S, \forall Ln_{ij}^S \notin L^N(\mathcal{G}_{i-1}) \right\}_{N_i^{LNS}} \\ & \cup \left\{ La_{ij}^S \mid \exists Ln_{ij}^S \in L^N(\mathcal{G}_{i-1}), W_j^{La_{ij}^S} \geq W_j^{La_{(i-1)j}^S} \right\}_{N_i^{LNS}} \\ & \cup \left\{ La_{(i-1)j}^S \mid \exists Ln_{ij}^S \in L^N(\mathcal{G}_{i-1}), W_j^{La_{ij}^S} < W_j^{La_{(i-1)j}^S} \right\}_{N_i^{LNS}} \end{aligned} \quad (6.30a)$$

$$\begin{aligned} I^A(\mathcal{G}_i) = & \left\{ Ia_{zj}^M \mid \forall Ia_{zj}^M \in \succ^M \right\}_{N_z^{INM-I}} \\ & \cup \left\{ Ia_{ij}^S \mid \forall Ia_{ij}^S \in \succ^S, \forall Pln_{ij}^S, CIn_{ij}^S \notin I^N(\mathcal{G}_{i-1}) \right\}_{N_i^{INS-I}} \\ & \cup \left\{ Ia_{ij}^S \mid \exists Pln_{ij}^S, CIn_{ij}^S \in I^N(\mathcal{G}_{i-1}), W_j^{Ia_{ij}^S} \geq W_j^{Ia_{(i-1)j}^S} \right\}_{N_i^{INS-I}} \\ & \cup \left\{ Ia_{(i-1)j}^S \mid \exists Pln_{ij}^S, CIn_{ij}^S \in I^N(\mathcal{G}_{i-1}), W_j^{Ia_{ij}^S} < W_j^{Ia_{(i-1)j}^S} \right\}_{N_i^{INS-I}} \end{aligned} \quad (6.30b)$$

Each arc conveys information regarding two operations and the order of their execution. Tree growing is thus performed based on the search and evaluation of arcs. Any change to an l -node will propagate upwards in the routing tree until the root node is reached, thus causing changes to all relevant i -nodes and affecting the tree structure as well. Therefore, in tree growing, l -node arcs are treated first and then i -node arcs. Moreover, tree growing operates on master l -node arcs and master i -node arcs first, and then selective l -node arcs and selective i -node arcs. Figure 6.9 illustrates the general procedure of tree growing.

Figure 6.9 Procedure of tree growing

For the master arc set of T_i , $\{La_{ij}^M | \forall La_{ij}^M \in \succ^M\}_{N_i^{LNM}}$ and $\{Ia_{ij}^M | \forall Ia_{ij}^M \in \succ^M\}_{N_i^{INM} - I}$, add their weights, $W_j^{La_{ij}^M}$ and $W_j^{Ia_{ij}^M}$, in G_{i-I} by the corresponding weight values in T_i . For $\{La_{ij}^S | \forall La_{ij}^S \in \succ^S\}_{N_i^{LNS}}$ of T_i , if they also can be found in G_{i-I} , then increase their weights $W_j^{La_{ij}^S}$ in G_{i-I} by the corresponding weight values in T_i . If they cannot be found in the selective l -node arc list of G_{i-I} , it may involve with one of the four situations illustrated in Figure 6.10: (a) nonexistence of l -nodes Ln_{ij}^S , (b) nonexistence of i -nodes In_{ij}^S , (c) nonexistence of both, and (d) existence of both but without a single path in between.

In the situations of Figures 6.10(a) and 6.10(d), both La_{ij}^S and Ln_{ij}^S are added to the respective arc sets of $\mathbf{G}_{i-l}$ directly. In Figures 6.10(b) and 6.10(c), however, the weights of La_{ij}^S , $W_j^{La_{ij}^S}$, need to be compared with those of relevant arcs in $\mathbf{G}_{i-l}$. First, search of La_{ij}^S in A_{AS} is performed. If it exists in A_{AS} , then increase its weight, $W_j^{La_{ij}^S}$, by the recorded weight

value in A_{AS} . The obtained new weight is used for comparison; otherwise, the original weight value is used in case that it does not exist in A_{AS} . In any case, if the weight value of La_{ij}^S is smaller, put La_{ij}^S in A_{AS} (or increase its weight only in case that it exists in A_{AS}); otherwise, add it to the selective l -node set of G_{i-1} and move the compared arcs from G_{i-1} to A_{AS} .

Figure 6.10 Different situations of an l -node arc in the member and generic trees

For example, in Figure 6.10(b), arc $e \succ f$ of T_i does not exist in G_{i-1} , and the connected l -node e exists in both. To determine which arc should be included in the generic routing in order to maintain a valid tree structure, weight comparison is performed. First, check $e \succ f$, and if it exists in A_{AS} , then increase its weight value by adding up the recorded weight value in A_{AS} . If it cannot be found in A_{AS} , then keep the original weight value. Then evaluate arc $e \succ b$ in G_{i-1} . If the weight value of $e \succ f$ is greater than that of $e \succ b$, then add $e \succ f$ to the selective l -node arc set of G_{i-1} . Meanwhile, arc $e \succ b$ is removed from G_{i-1} and added into A_{AS} , and arc $e \succ f$ is removed from A_{AS} if it exists in it. If the weight value of $e \succ f$ is smaller, then put it in A_{AS} (or increase the weight value only if it exist in

A_{AS}). Similarly in Figure 6.10(c), the weights of $c \succ b$ in T_i and $c \succ a$ in G_{i-1} need to be compared.

For the selective i -node arcs of T_i , $\{Ia_{ij}^S | \forall Ia_{ij}^S \in \succ^S\}_{N_i^{INS}-1}$, if they can also be found in G_{i-1} , increase their weights in G_{i-1} by the weight values in T_i . Otherwise there are four possible situations: (a) nonexistence of child i -node Cl_{ij}^S , (b) nonexistence of parent i -node Pl_{ij}^S , (c) nonexistence of both, and (d) existence of both. If both parent and child i -nodes exist, two more cases are further discerned: there is a single path between two nodes, or there is no a single path connecting them.

As in the case shown in Figure 6.11(c), neither of two nodes of Ia_{ij}^S can be found in the selective i -node set of G_{i-1} , in this situation, both nodes and arcs are directly added to the i -node set and the selective i -node arc set of G_{i-1} .

Figure 6.11 Different situations of an i -node arc in the member and generic trees

As shown in Figure 6.11(d), tree unification first checks Ia_{ij}^S in A_{AS} . If it can be found, its weight value is increased by the recorded weight value in A_{AS} . The new weight value is

used for comparison; otherwise, the original weight value recorded in T_i is used. Subsequently, the relevant arcs in G_{i-l} are compared with Ia_{ij}^S . After comparison, proper arc or node removal and addition operations are carried out based on the result of comparison. For example, in Figure 6.11(d.2), arc $c \succ b$ of T_i does not exist in G_{i-l} while both nodes b and c can be found in the i -node set of G_{i-l} . First A_{AS} is searched for $c \succ b$. Increase the weight value if it can be found; otherwise, keep its original weight value, and then compare the weights of $c \succ b$ and $c \succ f$ in G_{i-l} . In case that the weight value of $c \succ b$ is larger, add it into the i -node set of G_{i-l} (and remove it from A_{AS} if it exists), in the meantime move $c \succ f$ from the selective arc set of G_{i-l} to A_{AS} . If the weight of $c \succ b$ is smaller, then put it in A_{AS} (and increase its weight value if it has already been there). In Figure 6.11(d.1), weight comparisons are enacted among arcs $b \succ a$ and $b \succ d$ or $d \succ a$ of G_{i-l} . If the weight of $b \succ a$ is higher, both $b \succ d$ and $d \succ a$ are moved to A_{AS} . Meanwhile, $b \succ a$ is removed from A_{AS} if it exists. As shown in Figures 6.11(a) and 6.11(b), nodes that cannot be found in G_{i-l} are added into the corresponding sets first. Then operations on arcs proceed in the similar way to that of other situations.

Upon completion of the tree growing process, the formed GR consists of a generic routing structure and an additional arc set. Repeating the procedure, the GR for other clusters are obtained. Treating such formed GR of each cluster as member trees and applying the unification process to them leads to the GR for the entire process family. Similarly, the final GR includes a unified generic routing structure and an extended additional arc set resulted from these of each cluster. Due to the presence of selective arcs in the generic tree, the GR is by no means the union of all member trees. Addition and removal of certain arcs according to their weights guarantee that the resulted generic routing is the most common among individual routings in a process family.

6.4. Case Study

Test of the proposed data mining methodology for identifying generic routings for process families is carried out in the same company as the one in Chapter 4. In this case study, product and process data of one vibration motor family containing 30 product models are used. In this motor family, the component parts include “rubber holder” (rh), “weight” (wt), “frame” (fm), “bracket a” (ba), “bracket b” (bb), “terminal” (tl), “magnet” (mt), “magnet housing” (mh), “coil” (cl), “shaft” (st), “commutator” (ct), “tape” (tp), and “washer” (ws). For each component type, there are a number of variants catering for specific mobile phone requirements. Figure 6.12 shows the routings for producing two different motors. Details of 30 routings are given in Appendix B. In each routing tree, the nodes represent specific operations (manufacturing or assembly). The label of each node indicates the ID of the operation concerned. For example, “FmA2” indicates a particular assembly operation for producing the final motor, and “StM3” denotes a specific variant of shaft machining operation.

Figure 6.12 Two routing variants for producing vibration motors

6.4.1. Routing Similarity Measure

To measure similarity among the 30 routings, the number of *i*-nodes of each routing is identified, based on which the number of representation trees for each routing is calculated as 2^N , where *N* is the number of *i*-nodes. Table 6.1 lists the number of representation trees for each routing associated with the vibration motor family.

Table 6.1 Numbers of representation trees for each routing

Routing variants	Number of i-nodes	Number of representation trees
R1, R2, R3, R4, R5, R6, R7, R8	11	2048
R9, R10, R11, R12, R13, R14, R15, R16, R17, R18	10	1024
R19, R20, R21, R22, R23, R24, R25	9	512
R26, R27, R28, R29, R30	8	256

The SPSS® software is employed for analysis because it is able to analyze text data. Three attributes are used to describe the characteristics of each operation, including material, product, and resource types. In preparing data files for text mining, raw materials are described as material components of manufacturing operations. Figure 6.4 shows an example of part descriptions. Operation description data files are obtained by enumerating all operations contained in the 30 routings according to their types. Further extracting material component descriptions from these files generates both primitive component text files and compound component files. These files are input into SPSS for analysis.

Figure 6.13 shows the results of text analysis of the primitive component type “ba”, including the extracted keywords, i.e., attribute values describing “ba” variants, and their respective occurrence counts. Based on extracted information, the weights of relevant attributes are calculated, as shown in Table 6.2.

Table 6.2 Attributes and their relative importance

Attribute	Value Set	Weight
Shape	Square, round, rectangle, trapezoid, half-oval-rectangle	0.235
Color	Black, yellow, gray, white, blue	0.165
Material	ABS, acrylic, pthene, PVC, nylon	0.152
Weight	0.05g, 0.074g, 0.08g, 0.084g, 0.12g	0.224
Thickness	1.52mm, 1.85mm, 2.37mm, 3.04mm, 3.53mm	0.224

Figure 6.13 Extracted keywords and their occurrences for “ba” component type

For the set of attributes in Table 6.2, shape, color, and material are of nominal type while weight and thickness are numerical. To quantify each nominal attribute, a semantic scale is assigned to its specific instances based on domain knowledge. Table 6.3 shows the scaled values for all attribute instances.

Table 6.3 Similarity scales for nominal attributes of “ba”

Shape	Instance	Square	Round	Rectangle	Trapezoid	Half-oval-rectangle
	Scaled Value	0.28	0.35	0.27	0.44	0.53
Color	Instance	Black	Gray	White	Yellow	Blue
	Scaled Value	0.32	0.16	0.47	0.36	0.25
Material	Instance	ABS	PVC	Acrylic	Pthene	Nylon
	Scaled Value	0.56	0.21	0.39	0.62	0.45

Based on established semantic scales, attribute similarity measures are calculated using Eq. (6.7), and the result is shown in Table 6.4. When measuring attribute similarity, 0 is used as the smallest semantic value to indicate nonexistence of an attribute. Based on the results of attribute similarity, similarity measure of component “ba” among 30 routing variants is derived using Eq. (6.8). The result is presented in a matrix form as shown in Figure 6.14. The values in the matrix indicate the similarity of two “ba” variants in the corresponding column and row.

Table 6.4 Result of attribute similarity measure

Weight Similarity					Thickness Similarity				
1	0.20	0.25	0.28	0.58	1	0.09	0.24	0.43	0.57
0.20	1	0.05	0.08	0.38	0.09	1	0.14	0.34	0.48
0.25	0.05	1	0.03	0.33	0.24	0.15	1	0.19	0.33
0.28	0.58	0.03	1	0.30	0.43	0.34	0.19	1	0.14
0.58	0.38	0.33	0.30	1	0.57	0.48	0.33	0.14	1
Shape Similarity					Color Similarity				
1	0.13	0.02	0.30	0.47	1	0.34	0.32	0.09	0.15
0.13	1	0.15	0.17	0.34	0.34	1	0.66	0.43	0.19
0.02	0.15	1	0.32	0.49	0.32	0.66	1	0.23	0.47
0.30	0.17	0.32	1	0.17	0.09	0.43	0.23	1	0.23
0.47	0.34	0.49	0.17	1	0.15	0.19	0.47	0.23	1
Material Similarity									
1	0.56	0.27	0.10	0.18					
0.56	1	0.29	0.66	0.39					
0.27	0.29	1	0.37	0.10					
0.10	0.66	0.37	1	0.27					
0.18	0.39	0.10	0.27	1					

1	.81	0	.88	.84	1	.80	.88	1	.19	.17	.13	0	.21	0	.11	.21	0	.19	.27	1	.11	.19	0	.21	.22	0	.18	0	.15	
.81	1	0	.83	.85	.81	.89	.83	.81	.10	.14	.17	0	.18	0	.13	.21	0	.10	.10	.81	.20	.10	0	.18	.17	0	.21	0	.13	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
.88	.83	0	1	.78	.88	.82	.90	.88	.17	.19	.15	0	.23	0	.11	.19	0	.17	.17	.88	.13	.17	0	.23	.31	0	.20	0	.13	
.84	.85	0	.78	1	.84	.86	.85	.84	.05	.19	.30	0	.15	0	.13	.15	0	.05	.13	.84	.25	.12	0	.15	.14	0	.18	0	.08	
1	.81	0	.88	.84	1	.80	.88	1	.19	.17	.13	0	.21	0	.11	.21	0	.19	.27	1	.11	.19	0	.21	.22	0	.18	0	.15	
.80	.89	0	.82	.86	.80	1	.82	.80	.16	.15	.18	0	.19	0	.20	.20	0	.09	.09	.80	.21	.09	0	.26	.18	0	.23	0	.12	
.88	.83	0	.90	.85	.88	.82	1	.88	.17	.27	.23	0	.23	0	.11	.27	0	.17	.17	.88	.13	.24	0	.23	.24	0	.20	0	.13	
1	.81	0	.88	.84	1	.80	.88	1	.19	.17	.13	0	.21	0	.11	.21	0	.19	.27	1	.11	.19	0	.21	.22	0	.18	0	.15	
.19	.10	0	.17	.05	.19	.16	.17	.19	1	.76	.73	0	.15	0	.15	.15	0	.25	.25	.19	.05	.25	0	.22	.16	0	.12	0	.12	
.17	.14	0	.19	.19	.17	.15	.27	.17	.76	1	.87	0	.28	0	.12	.22	0	.11	.18	.17	.19	.11	0	.21	.20	0	.17	0	.08	
.13	.17	0	.15	.30	.13	.18	.23	.13	.73	.87	1	0	.17	0	.15	.18	0	.08	.08	.13	.23	.15	0	.17	.24	0	.21	0	.16	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
.21	.18	0	.23	.15	.21	.19	.23	.21	.15	.28	.17	0	1	0	.77	.84	0	.80	.87	.21	.15	.15	0	.25	.24	0	.22	0	.12	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
.11	.13	0	.11	.13	.11	.20	.11	.11	.15	.12	.15	0	.77	0	1	.80	0	.73	.73	.11	.21	.08	0	.27	.11	0	.23	0	.05	
.21	.21	0	.19	.15	.21	.20	.27	.21	.15	.22	.18	0	.84	0	.80	1	0	.80	.80	.21	.15	.15	0	.19	.18	0	.22	0	.15	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
.19	.10	0	.17	.05	.19	.09	.17	.19	.25	.11	.08	0	.80	0	.73	.80	0	1	.90	.19	.05	.25	0	.15	.16	0	.12	0	.12	
.27	.10	0	.17	.13	.27	.09	.17	.27	.25	.18	.08	0	.87	0	.73	.80	0	.90	1	.27	.05	.25	0	.15	.16	0	.12	0	.12	
1	.81	0	.88	.84	1	.80	.88	1	.19	.17	.13	0	.21	0	.11	.21	0	.19	.27	1	.11	.19	0	.21	.22	0	.18	0	.15	
.11	.20	0	.13	.25	.11	.21	.13	.11	.05	.19	.23	0	.15	0	.21	.15	0	.05	.05	.11	1	.70	0	.88	.14	0	.26	0	.08	
.19	.10	0	.17	.12	.19	.09	.24	.19	.25	.11	.15	0	.15	0	.08	.15	0	.25	.25	.19	.70	1	0	.80	.16	0	.12	0	.12	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
.21	.18	0	.23	.15	.21	.26	.23	.21	.22	.21	.17	0	.25	0	.27	.19	0	.15	.15	.21	.88	.80	0	1	.24	0	.30	0	.12	
.22	.17	0	.31	.14	.22	.18	.24	.22	.16	.20	.24	0	.24	0	.11	.18	0	.16	.16	.22	.14	.16	0	.24	1	0	.86	0	.85	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
.18	.21	0	.20	.18	.18	.23	.20	.18	.12	.17	.21	0	.22	0	.23	.22	0	.12	.12	.18	.26	.12	0	.30	.86	0	1	0	.77	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
.15	.13	0	.13	.08	.15	.12	.13	.15	.12	.08	.16	0	.12	0	.05	.15	0	.12	.12	.15	.08	.12	0	.12	.85	0	.77	0	1	0

Figure 6.14 Similarity matrix for primitive component “ba”

For “Bracket Assembly” operation, one primitive components, “tl”, and one compound component, “bracket a/b”, constitute its material components while compound component “bassy” is the product component of this operation. Based on Eq. (6.9), compound component similarity is derived for “bassy”. Figure 6.15 gives the result of product component similarity of “bassy” among 30 routings.

Figure 6.15 Similarity matrix for compound component “bassy”

Resource similarity measure proceeds with text analysis in a similar fashion, where machine and setup are nominal variables, and cycle time is numerical. Semantic scales for nominal attributes machine and setup are assigned by the company’s production engineers. Applying Eqs. (6.10-6.13), the resource similarity matrix is obtained, as shown in Figure 6.16.

Figure 6.16 Resource similarity matrix for “Bracket Assembly” operation

Compiling results of component and resource similarity measures, operation similarity is derived using Eq. (6.2). Figure 6.17 shows the similarity matrix for “Bracket Assembly” operation. Similarity calculation of all operation variants of same types in the motor family is conducted in the same way. With all the operation similarity matrices available, further applying Eqs (6.1) and (6.14), the normalized node content similarity measures are calculated, as given in Figure 6.18.

Figure 6.17 Operation similarity matrix for “Bracket Assembly”

Figure 6.18 Node content similarity matrix for 30 routings

To measure tree structure similarity, representation trees are generated for each routing, and then tree edit graphs are established for every paired trees in accordance with the number of i -node contained in each routing tree. The result of pairwise distance measures of 30 routing trees is shown Figure 6.19. Applying Eqs. (6.16) and (6.17) to these distance measures, the tree structure similarity matrix is calculated, as given in Figure 6.20. Finally, compiling node content similarity (Figure 6.18) and tree structure similarity (Figure 6.20), the normalized pairwise similarity measures of 30 routings are obtained, as given in Figure 6.21.

Figure 6.19 Tree structure distance matrix of 30 routings

Figure 6.20 Tree structure similarity matrix of 30 routings

Figure 6.21 Routing similarity matrix

6.4.2. Routing Clustering

By its property, the routing similarity matrix is itself a fuzzy compatible matrix. Applying the max-min composition to the routing similarity matrix in Figure 6.21, a fuzzy equivalence matrix is obtained as shown in Figure 6.22. Based on domain knowledge on clustering, a threshold level of 0.85 is decided. Accordingly, the λ -cut matrix is obtained as shown in Figure 6.23. The netted graph is developed as shown in Figure 6.24, based on which the routing clusters are derived. Table 6.5 gives the result of routing clustering with four clusters identified.

Figure 6.22 Routing fuzzy equivalence matrix

Figure 6.23 The λ -cut of routing fuzzy equivalence matrix ($\lambda=0.85$)

Figure 6.24 Netted graph for the λ -cut in Figure 6.23

Table 6.5 Routing clusters and respective members

ROU Cluster	ROU Variants
RC1	R1, R3, R10, R13, R14, R17, R20, R22, R25
RC2	R2, R4, R5, R6, R7, R8, R9, R11, R16, R18
RC3	R23, R26, R27, R28, R29, R30
RC4	R12, R15, R19, R21, R24

Inclusion Conditions:

- 1) IF $Bassy_3^*$ AND $Bassy_{11}^*$
THEN $Tl\ Machining_i^* > BracketAssy\ Assembly_i^* = 0$
- 2) IF $CoilAssy_3^*$ AND $CoilAssy_{16}^*$
THEN $TP\ Shaping_i^* > CoilAssy\ Assembly_o^* = 1$

Execution Rule:

- 1) IF $Mtassy_8^*$ AND $Rtassy_{14}^*$
THEN $FassyAssemblyW_5^*$ AND
 $FassyAssemblyS_2^*$ AND
 $FassyAssemblyT_{11}^*$

Figure 6.25 The Generic routing of a motor family

6.4.3. Routing Unification

For each routing cluster, one GR is formed by tree growing. The GR of the entire family (i.e., 30 individual routings) is obtained by unifying the four GRs for the four clusters. Figure 6.25 presents the final GR of the entire family, which is represented using UML. As shown in the figure, not all operation types are necessary for producing each motor variant. For example, “wt” is purchased rather than manufactured in-house for some customer orders, and thus the corresponding “wt” machining operation manifests itself as a type of selective operation.

6.5. Performance Evaluation

The performance of generic routing identification with respect to the mass production efficiency of product families entails the specification of a proper value of similarity threshold of the λ -cut. Essentially, it gives rise to a tradeoff issue of generic routing granularity inherent in mass customization production (Jiao et al., 2007). With a large (small) value of λ -cut, fewer (more) generic routings will be identified. These generic routings affect the downstream performance of process platform-based production configuration. From the economic viewpoint, the cost of introducing more product and process families (i.e., finer routing identification) and its contribution to customer satisfaction should reach a balance at the right level of aggregation of the product and process platforms. If the differentiation of product families is too spread or at too low a level of aggregation, such as at the nuts and bolts level, then the number of routing elements may become too many, and product fulfillment becomes difficult to leverage investments. To the contrary, if routings are identified at a very high level, such as complete assemblies, then the repetition may not be sufficient to exploit the economy of scale in production.

A proper granularity can normally be determined by assessing the performance of the product and process platforms in accordance with the resulted generic routings. Jiao and

Zhang (2005a) apply conjoint analysis and choice models to evaluate customer perceived utilities of diverse offerings of product families. On the other hand, the development of product and process platforms embodies a type of fixed costs (Meyer and Lehnerd, 1997). Based on the work of the two groups, a performance measure of λ -cut, Ψ^λ , is introduced for evaluation.

$$\Psi^\lambda = \frac{E[U]}{C^F}, \quad (6.31)$$

where $E[U]$ denotes the expected utility of product families as perceived by the customers, which is determined based on a planning framework in (Jiao and Zhang, 2005a), and C^F stands for the fix cost of the product and process platforms corresponding to the identified generic routing. Jiao and Tseng (2004) posited the rationale of justifying cost implications of the product and process platforms based on process variations. Adopting their model, this research employs a process capability index to measure the above fixed cost, as follows:

$$C^F = \beta^F e^{\frac{1}{PCI}} = \beta^F e^{\frac{6\sigma}{USL-LSL}}, \quad (6.32)$$

where β^F is a constant indicating the average dollar cost per variation of process capabilities, USL , LSL , and σ are the upper specification limit, lower specification limit, and standard deviation of part-worth cost estimates corresponding to individual routing clusters, respectively. The part-worth cost estimates are determined using a pragmatic approach based on standard time estimation in (Jiao and Tseng, 1999).

To analyze the sensitivity of routing identification, a total of 19 clustering runs are generated by changing λ value from 0.05 to 0.95 with an increment of 0.05. Using process data of vibration motors in Jiao et al., (2007) and utility evaluation data in Jiao and Zhang, (2005a), the result of the sensitivity analysis with respect to the similarity threshold values is obtained, as shown in Figure 6.26. The performance measure in Eq. (6.31) is presented as a

normalized comparison. The result clearly shows that a λ value of 0.85 yields the best performance of mining in regard to routing identification for this problem.

Figure 6.26 Sensitivity analysis of routing identification

The performance of the proposed data mining approach is further tested by benchmarking with the traditional manual approach performed by domain experts. Also considered is the performance with respect to different data sizes. Five different sets of routing data are used, including 30 (the base case used in the case study), 60, 100, 150 and 220 product models produced in the company. Similar to the previous sensitivity analysis, the performance measure in Eq. (6.31) is used.

Figure 6.27 Performance comparison of the data mining and manual approaches

Figure 6.27 shows the results of testing in terms of normalized comparison. As shown in the figure, the data mining approach outperforms the manual method overall. As the data size increases, the advantage of the data mining approach over the manual method becomes

more evident. While the data mining approach performs better with more routing data, the performance of the manual method does not keep pace with the increase in data size and even deteriorates when given a large amount of data because of the complexity. This may verify the fact that data mining usually works well when large amounts of data are available. To the contrary, human intelligence works with limited amount of data and always gets lost when facing overwhelming amount of data.

6.6. Summary

Process platform construction is addressed in this chapter. Underlying a process platform is the generic routing structure of the associated process family. The problem of process platform construction can thus be converted to that of generic routing formation. With consideration of the large amount of easily accessible product and process data as well as advanced data mining tools, a systematic data mining methodology is developed to identify generic routings from historical data. Generic routing identification with the use of available data mining tools and historic data is advantageous to process platform construction in that both the cost and time can be reduced through the reuse of proven elements. In addition, company's existing knowledge can be sustained and maintained by storing and mining historical data. Furthermore, the proposed methodology allows incorporating experts' experiences into the projection of routing planning patterns from historical data, thereby enhancing the ability to explore and utilize domain knowledge more effectively.

In the methodology, generic routings are identified through a routing similarity measure, routing clustering, and routing unification. To accomplish this, a number of techniques, including text mining, tree matching, bipartite matching, distance measure, normalization, fuzzy clustering, and tree unification are adopted in the methodology. The reported application case of vibration motors demonstrates the feasibility of the proposed methodology. Performance evaluation of the data mining methodology in terms of sensitivity

analysis of similarity threshold values helps determine the optimal number of generic routings to be constructed for process families. In the next chapter, an association rule mining approach will be presented to address variety coordination in process platforms.

CHAPTER 7:

VARIETY COORDINATION BASED ON ASSOCIATION RULE MINING

Variety coordination attempts to specify necessary production variations quickly while eliminating unnecessary variations in relation to design changes in customized products. It assists companies in maintaining stable production. This is accomplished by identifying mapping relationships between product and process variety exhibited by available product and process data. Due to the unstructured nature inherent in mapping relationships between product variety in the product domain and process variety in the process domain as well as the complexities in dealing with two domains simultaneously, it is difficult to determine the mapping relationships without proper tools. As one of the most practical and beneficial data mining techniques, association rule mining can discover the implicit yet significant associations among objects being analyzed. Thus, association rule mining is adopted in this research to uncover associations between historical product and process data and identify mapping relationships between the two forms of variety. In the following, the related work regarding association rule mining is reviewed first.

7.1. Association Rule Mining Techniques

As one of the major forms of data mining, association rule mining is perhaps the most common form of knowledge discovery in unsupervised learning systems (Chen et al., 1996). It may be one of the most practical and beneficial data mining techniques among all in that implicit yet significant associations among objects being analyzed can be discovered through rule mining and thus assists people in making right decisions. Association rules are produced by finding the interesting associations or correlations among a large set of data items. The

flexibility of association rule induction lies in its capability to deal with qualitative data that cannot be treated by traditional operations research methods.

Ever since it was introduced into the data mining community by Agrawal et al. (1993), association rule mining has received a great deal of attention, partly due to its apparent utility and partly due to the research challenges it presents. Association rule mining can be applied in a wide range of applications domains, such as sales and marketing (Chen et al. 1996), engineering design (Agard and Kusiak, 2004), manufacturing (Chen, 2003), product portfolio planning (Jiao and Zhang, 2005a), Web mining (Chen et al., 1998; Cooley et al., 1999), recommender systems (Lin et al., 2002), and document management (Zhou, 2003).

The earliest form of association rule mining is *market basket analysis* (Agrawal et al., 1993; Chen et al., 1996), in which the basket consists of a set of items purchased by a customer on a single store visit. The discovered customer buying patterns that are previously unknown and hidden in the large databases of sales data benefit business corporations enormously in many aspects, including store layout for example.

According to the type of databases, association rule mining can be categorized as quantitative association rule mining and Boolean association rule mining (Lin et al., 2002). While the former mines databases of numerical and categorical attributes, the latter (the basic one) is applied to Boolean attribute databases. Typical solutions to quantitative association rule mining are to 1) partition each numerical attribute into a set of disjoint intervals, 2) interpret each interval as an item, and 3) apply standard Boolean association rule mining. Li et al. (1999) introduced an adaptive partitioning method based on repeatedly merging smaller intervals into larger ones so as to improve mining efficiency compared with others work, e.g., (Dougherty et al., 1995; Fayyad and Irani, 1993).

Since the introduction of the Apriori algorithm (Agrawal and Srikant, 1994), there has been sustained interest in discovering new rule mining algorithms that could perform more efficiently. Incremental algorithms were presented in (Han et al., 2000; Cheung et al., 1997),

and algorithms that support dynamic support thresholds were introduced by Aggarwal and Yu (1998) and Hidber (1999). Focusing on the speed of algorithms, Das et al. (2001) discussed the Rapid Association Rule Mining algorithm that achieved significant speed-ups against the Aprior algorithm.

A number of association rule mining applications have been reported recently. Chen (2003) applied association rule mining to the problem of forming cellular manufacturing systems. In his cell formation approach, association rule induction was used to find associations among machines that can be grouped in same manufacturing cells. Cooley et al. (1999) and Chen et al. (1998) employed association rule mining to find path traversal patterns that can facilitate the best design and organization of Web pages. Fu et al. (2000) developed a system for recommending Web pages. The Aprior algorithm was adopted in his system to mine association rules over users' navigation histories. Jiao and Zhang (2005a) applied association rule mining to find correlations among customer needs and design attributes in a market segment so as to provide a proper product portfolio to markets. Agard and Kusiak (2004) adopted association rule mining at the early stage of product family design, i.e., mining associations between functional requirements and design options based on gathered customer data. For improving the efficiency in editing computer-aided engineering documents by reusing the same document content, Zhou (2003) used association rule mining to obtain the relations among document structures and items contained in the structures. For a document to be edited, the items that may be contained will be known quickly and edited accordingly based on the rule mining results.

While most work in association rule mining deals with objects (or items) from a single domain, there are some cases that require mining associations between objects of different domains. One work has been reported to address such problems of two domain items in a single mining (Jiao and Zhang, 2005b). The proposed rule mining approach in this dissertation is to mine associations between items from two domains, i.e., the product and

process domains. Further, the focus is on the application of association rule mining rather than the improvement of rule mining algorithms. In the next section, traditional association rule mining is presented.

7.2. Association Rule Mining Formulation

The basic problem of mining association rules is introduced by Agrawal and Srikant (1994). Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of literals, called items. Let TDB be a database of transactions, where each transaction, T , is a set of items such that $T \subseteq I$, and each transaction is associated with an identifier, called TID . Given $Z \subseteq I$, a transaction T contains Z if and only if $Z \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$. The association rule $X \Rightarrow Y$ holds in TDB with confidence c if $c\%$ of the transactions in TDB that contain X also contain Y . This is taken to be a conditional probability, $P(y/x \mid \forall x \in X, \forall y \in Y)$. The association rule $X \Rightarrow Y$ has support s in TDB if $s\%$ of the transactions in TDB contain X and Y . The support is taken to be a probability, $P(x \wedge y \mid \forall x \in X, \forall y \in Y)$.

While the confidence denotes the strength of implication, the support indicates the frequencies of the occurring patterns in the rule. Given a minimum confidence threshold, min_conf , and a minimum support threshold, min_sup , the problem of mining association rules becomes a search for all the association rules whose confidence and support are larger than the respective thresholds. Based on whether the thresholds (min_conf and min_sup) can be met, association rules are distinguished between strong rules and weak rules.

A set of items is referred to as an *itemset*. An *itemset* that contains k items is called a k -itemset. Given a minimum support threshold, min_sup , an *itemset* is called large if its support is no less than min_sup . Association rule mining involves a two-step process (Agrawal et al., 1993): (1) discover all large itemsets whose support is larger than the

predetermined minimum support threshold - Itemsets with minimum support are called frequent itemsets - and (2) generate strong association rules from the large itemsets.

The most crucial factor affecting the performance of mining association rules lies in the first step. After the large itemsets are identified, the corresponding association rules can be derived in a straightforward manner. Efficient counting of large itemsets is hence the focus of most prior studies on algorithms for mining association rules.

7.3. Association Rule Mining in Variety Coordination

As observed in the traditional association rule mining, a *TDB* contains items belonging to the same domain. The intention of this research is to discover the mapping relationships between product variety in the product domain and process variety in the process domain rather than interrelations in any single domain. Therefore, items in a *TDB* in the proposed approach are from two different domains: the product and process domains.

Products are built from items, which may be parts or assemblies. Accordingly, mining rules in the proposed approach is conducted within each item family. Thus, a *T* in the *TDB* of an item family corresponds to a particular item variant, either of a part type or an assembly one. While a part is specified by its design parameters along with their specific value instances, i.e., parameter value pairs, and based on these design parameter value pairs, its process elements are determined; an assembly is uniquely formed and described accordingly by a number of particular child items of part types or/and assembly types, and the selection of its process elements is influenced by these child items. Therefore, mining association rules for parts and assemblies is approached differently.

7.3.1. Association Rule Mining for Parts

For a given *TDB* of a part family *B* with *n* variants, every *T* relates to a specific part variant B_i^* . The total number of transactions in the *TDB* is thus the same as that of part

variants in the family, i.e., n . The product-related items in a T_i are a list of design parameter value pairs, $T_i = \{B_i^*\}_n \sim \{P_s V_t\}_{p \times v}$, where $P_s V_t$ represents the t -th value instance of the s -th design parameter assumed by B_i^* , p is the total number of design parameters in the part family, and v is the total number of possible value instances, which are assumed by all variants in the family, of the s -th design parameter, $v \leq N$. The process data of a part include the set of operations (O), each of which is described by its type (α), machine (M), tooling (T), fixture (F), and cycle time (C). The process-related items of T_i in the transaction TDB is formulated as:

$$T_i = \{B_i^*\}_n \sim \{O_a\}_p = \{\langle \alpha_a, M_a, T_a, F_a \rangle\}_p, \quad (7.1)$$

where p is the total number of operations for producing B_i^* . The cycle times are not included in a TDB , because their values are influenced by the selection of other process elements.

7.3.2. Association Rule Mining for Assemblies

While each part is made from a type of raw material, an assembly is formed by joining several child items. The assembly variants $\{A_i^*\}_m$ of an assembly family A differ from one another in their unique configuration of various child items. These variations lead to the selection of different process elements.

A T_i in a TDB of A corresponds to a specific variant. Similarly, the total number of T in the TDB equals to that of assembly variants in the family. The product items of each T_i relates to child items of the corresponding A_i^* , i.e., $T_i = \{A_i^*\}_m \sim \{I_x V_y\}_{i \times v}$, where $I_x V_y$ represents the y -th variant of the x -th type child item, i is the total number of child item types in the assembly family, and v is the total number of item variants of the x -th type. Likewise, the process-related items of T_i are as follows:

$$T_i = \{A_i^*\}_m \sim \{O_t\}_q = \{\langle \alpha_t, M_t, T_t, F_t \rangle\}_q, \quad (7.2)$$

where q is the total number of assembly operations for making A_i^* . The cycle times of each operation are not included for the same reason as that for part association rule mining.

7.3.3. Procedure for Association Rule Mining

The procedure for mining association rules involves five steps as follows:

Step 1: Data preprocessing. This step attempts to preprocess raw data into a proper format for association rule mining. The raw data should be processed based on the identified product and process families. For parts that are manufactured in-house, their design parameter value pairs that influence the selection of process elements are encoded in the target data set. For assemblies, their immediate child items are included in the target data set. Also encoded into the target data set are the associated process elements for all items either of part types or assembly types. Then the sorted target data are arranged in *TDBs*, each of which is for an item family. Table 7.1 gives an example of *TDBs* for a part family and an assembly family, respectively. While in *TDBs* process-related item data are of exactly the same kinds, i.e., α , M , T , and F , product-related item data are different for parts and assemblies. More specifically, product-related items in a *TDB* for a part family are design parameter value pairs (PVs), whereas those in a *TDB* for an assembly family are child components (IVs) of each member.

Table 7.1 Transaction databases of item families

(a) TDB of a part family		
ID	Transaction	Items {Product; Process}
1	B_1	$\{PV_1, PV_3, PV_1, PV_6; \alpha_1, M_1, T_1, F_1, \alpha_2, M_2, T_2, F_2, \alpha_3, M_3, T_3, F_3\}$
2	B_2	$\{PV_2, PV_1, PV_1, PV_6; \alpha_1, M_1, T_1, F_1, \alpha_2, M_2, T_2, F_2, \alpha_4, M_4, T_4, F_4\}$
3	B_3	$\{PV_1, PV_3, PV_2, PV_2; \alpha_1, M_1, T_1, F_1, \alpha_3, M_3, T_3, F_3\}$
4	B_4	$\{PV_4, PV_3, PV_4, PV_1; \alpha_2, M_2, T_2, F, \alpha_3, M_3, T_3, F_3\}$
(b) TDB of an assembly family		
ID	Transaction	Items {Product; Process}
1	A_1	$\{IV_2, IV_1, IV_2, IV_1; \alpha_1, M_1, T_1, F_1, \alpha_2, M_2, T_2\}$
2	A_2	$\{IV_2, IV_2, IV_1, IV_3; \alpha_1, M_1, T_1, F_1, \alpha_2, M_2, T_2, F_2, \alpha_3, M_3, T_3, F_3\}$
3	A_3	$\{IV_1, IV_2, IV_1; \alpha_1, M_1, T_1, F_1, \alpha_3, M_3, T_3, F_3\}$
4	A_4	$\{IV_4, IV_1, IV_4; \alpha_1, M_1, T_1, F_1, \alpha_2, M_2, T_2\}$

Step2: Mining association rules. The inclusion of data from two domains in a *TDB* leads to a general form of association rules for a part family, as follows:

$$X \Rightarrow Y \quad [Support = s\%, \quad Confidence = c\%], \quad (7.3)$$

where $Y = \{\alpha_i\}_a \wedge \{M_j\}_b \wedge \{T_s\}_c \wedge \{F_t\}_d$, the values of a , b , c and d may or may not be the same for the four kinds of process elements, $X = \{P_i V_j\}_{x \times y} | x \leq p, y \leq v$, and $s\%$ and $c\%$ refer to the support and confidence for the rule, respectively. Similar to the traditional association rule mining, $s\%$ and $c\%$ are calculated based on the following,

$$s\% = \frac{Count(X \wedge Y)}{Count(TDB)} \times 100\%, \quad (7.4)$$

$$c\% = \frac{Count(X \wedge Y)}{Count(X)} \times 100\%, \quad (7.5)$$

where $Count(X \wedge Y)$ is the number of transactions in the *TDB* of the part family that contain all design items in X and all process items in Y , $Count(TDB)$ is the total number of transactions in the *TDB*, $Count(X)$ is the number of transactions in the *TDB* that contain all design items in X . Because each T_i in the *TDB* represents a specific part B_i^* in the part family B , $Count(TDB) = n$. In addition, the two sets X and Y embody two non-empty subsets of I and E of the part family. The rule in Eq. (7.3) indicates that the occurrence of product data in X , more specifically design parameter value pairs, will most likely (at a $s\%$ support and a $c\%$ confidence) relate to that of process elements in Y .

The general rule of assembly transactions is exactly the same as that of parts in Eq. (7.3). The difference is that in X , the items are child items of the assembly family. Besides, the implication of the rule is that the occurrence of child items in X will most likely (at a $s\%$ support and a $c\%$ confidence) associate with that of process elements in Y . The Apriori algorithm (Agrawal and Srikant, 1994) is adopted in this research owing to its simplicity and strength.

The working principle of the algorithm includes two steps. In the first step, it finds all large itemsets, i.e., sets of items that have transaction support no less than the pre-defined minimum support (min_sup). In the second step, it derives the association rules from these identified itemsets. In the algorithm, the generation of rules is influenced by the specification of two pre-determined thresholds, namely the min_sup and the minimum confidence (min_conf).

Step 3: Generalization of generic mapping rules. The identified associations address the specific mapping relationship from products to processes at a lower and concrete level. They assist in determining particular process elements for a particular product item, but they cannot explain the connections between product and process at a higher and abstract level, for example, the class level. The specification of these generic associations is essential to the subsequent determination of specific process elements. This step tackles this problem and attempts to generate a generic relationship, $M^{(I^{G^*}, E^G)}$, between a generic product item, I^{G^*} , and a set of generic process items, E^G , i.e., $M^{(I^{G^*}, E^G)} = \langle I^{G^*}, E^G \rangle$. It is accomplished by generalizing previous mined specific association rules.

In a manufacturer's manufacturing resource database, every group of similar manufacturing resource variants, e.g., specific machines of the same type, are recorded in one data table. These data tables are used in this methodology to generalize generic process elements, e.g., generic machines, from specific variants in the mined rules. Since product-related items in the identified association rules of parts and assemblies are different, the generalization of generic mapping rules for generic parts and generic assemblies is solved differently. The procedure for part families is as follows.

Step 1: List all mined association rules from a *TDB* in a table;

Step 2: Refer to the existing databases of manufacturing resources and replace each specific resource variant in the body (i.e., consequence) of a rule with its type;

Step 3: After the replacement, distribute these rules that have exactly the same body into a group and count the total number of rules in each group;

Step 4: For each group, do the following:

(4.1) Check the dimensional parameters and these parameters whose values are numbers in the head (i.e., condition) of each rule;

(4.2) Specify the value range of each of these parameters, i.e., the lowest value and the highest value of each parameter contained in the group;

(4.3) Replace specific parameter value pairs in the head of each rule with the same parameters coupled with the identified value ranges;

(4.4) Keep other categorical parameter value pairs unchanged;

(4.5) Combine the heads of different rules in the group using the OR relation and keep the body unchanged to form one unified generic rule; and

(4.6) Specify the selection conditions for the unified generic rule, for example, “*If the material is steel or alloy, and the width of the inner hole is less than 25mm and greater than 10mm, and ..., then the A type machine and B type fixture must be selected*”; and

Step 5: Compute the occurrence frequency of each generic rule as the percentage of the generalized rules among the total number of rules in the table.

The generalization of generic association rules between generic assemblies and generic process elements follows a similar procedure except difference in Steps (4) and (5), as follows:

Step 4: For each group, do the following:

(4.1) Check the child items in the head of each rule;

(4.2) Replace item variants with the corresponding generic items in the heads of all rules;

(4.3) Combine heads of rules in the group using the OR relation (note: same generic items are only documented once), and keep the body unchanged to form one unified generic rule; and

(4.4) Specify the selection conditions for each group, for example, “*If child item of type A or B and child item of type C, D and E are used, then operation type P and machine type M must be selected*”; and

Step 5: Compute the occurrence frequency of each generic rule as the percentage of rules in the corresponding group among the total number of rules.

The generalization yields a list of generic association rules between generic product items and the associated generic process elements, along with selection conditions for each generic rule and the occurrence frequencies. While the specific mined rules relate specific product data to process data, the generalized generic rules associate generic product items with generic process items. Thus variety coordination in process platforms is realized by the association rules of both specific and generic process elements. Both the specific association rules and generic ones are used to specify process elements for given product items. In conjunction with generic process structures, these mapping rules are used to assist in the generation of optimal routings that take advantage of repetitions in manufacturing and production.

7.4. Case Study

The proposed association rule mining approach to uncovering mapping relationships between product and process data has been tested by using the same vibration motor data in the previous chapters. As shown in Figure 5.10(b), three main assemblies forming the final motor are an armature assembly, a frame assembly, and a bracket assembly. Each of the three assemblies can be further broken down into several parts. Association rule mining has been

applied to these parts and assemblies to obtain the rules between product and process data in the motor item family.

For this application, Magnum Opus (Version 2.0.2, <http://www.rulequest.com/>), a data mining tool, is employed. Magnum Opus provides five association metrics: leverage, lift, strength, coverage and support, each of which is supported by a search mode. The case study only uses the support and strength modes for the handling of support and confidence measures, respectively. This is because the coverage, lift, and leverage criteria are not considered in the Apriori algorithm. The strength in Magnum Opus is equal to the confidence in the Apriori algorithm and the support is the same as the support in the Apriori algorithm. The search guarantees that only those rules with the highest values on the specified metric are found according to user-specified search settings.

A motor family is selected randomly. Then transaction databases are prepared for each item family. Table 7.2 and Table 7.3 show two transaction databases for *bracket a* family and the *final motor* family, respectively. Because Magnum Opus has the requirements on the input files, all data are extracted from each transaction database and are input as a text file into Magnum Opus.

Table 7.2 Transaction database for a part family

Transaction IDs	Bracket a Variants	Design Parameter Value Pairs	Process Elements
001	Ba1	Material (ABS), Shape (Square), Thickness (1.67mm), Number of fusing holes (1), Length (3.05mm)	InjectorHS35, LocatorIII, AdjustorA, TrayTV, Pre-alignment JigTI
002	Ba2	Material (Polythene), Shape (Half-Oval-Rectangle), Thickness (1.67mm), Outer diameter (4.32mm), Number of fusing holes (3), Length (3.8mm)	InjectorLS507, LocatorIII, AdjustorDE, TrayU, Pre-alignment JigXS
003	Ba3	Material (Round), Shape (Round), Thickness (0.94mm), Number of fusing holes (5), Outer diameter (4.32mm)	InjectorN1044, LocatorIV, AdjustorA, TrayU, Pre-alignment JigRS
004	Ba4	Material (Nylon), Shape (Trapezoid), Thickness (1.67mm), Outer diameter (4.82mm), Number of fusing holes (5), Length (4.78mm)	InjectorNII11, LocatorIII, AdjustorDE, TraySS, Pre-alignment JigSSII
...	...	...	...
0015	Ba15	Material (ABS), Shape (Round), Thickness (1.30mm), Outer diameter (4.82mm), Number of fusing holes (3)	InjectorHS35, LocatorAP, AdjustorP, TraySS, Pre-alignment JigRS

Table 7.3 Transaction database for an assembly family

Transaction IDs	Vibration Motor Variants	Specific Child Components	Process Elements
001	VM1	Armature assembly variant1, Frame assembly variant1, Bracket assembly variant1, Weight variant1	Frame caulking machineHSS, Armature insertion machine1088, Weight caulking machine50A, Weight caulking headII, Bracket holderV, Weight sitting jigII, Insertion jigI
002	VM2	Armature assembly variant1, Frame assembly variant2, Bracket assembly variant1, Weight variant3	Frame caulking machineLPS, Armature insertion machine1088, Weight caulking machineB36, Weight caulking headIV, Bracket holderV, Weight sitting jigIV, Insertion jigI
003	VM 3	Armature assembly variant1, Frame assembly variant1, Bracket assembly variant2, Weight variant2	Frame caulking machineHSS, Armature insertion machine1088, Weight caulking machine50A, Weight caulking headXII, Bracket holderIII, Weight sitting jigXII, Insertion jigI
004	VM 4	Armature assembly variant1, Frame assembly variant4, Bracket assembly variant2, Weight variant3	Frame caulking machineTVI, Armature insertion machine1088, Weight caulking machineB36, Weight caulking headIV, Bracket holderIII, Weight sitting jigIV, Insertion jigI
...	...	...	...
0019	VM 19	Armature assembly variant8, Frame assembly variant5, Bracket assembly variant4, Weight variant7	Frame caulking machineBXX, Armature insertion machineMS33, Weight caulking machineC01, Weight caulking headLL2, Bracket holderA, Weight sitting jigCX, Insertion jigVII

In the case study, the maximum number of associations is set to be 10,000 to make sure that the association rules can be completely explored. Figure 7.1 gives the setting of search modes and their metrics as well as the rule mining process in Magnum Opus for the extracted text file from the transaction database in Table 7.2 for bracket a family. The minimum leverage, minimum lift, minimum coverage, minimum strength and minimum support are set as 0, 0, 0 (default value required by the system), 0.2, and 0.2, respectively. Mining associations for other part and assembly families is conducted in the same manner.

Figure 7.2 gives the mining results for the transaction database in Table 7.2. The front small figure shows the statistics of the system setting and the number of mined rules; the back figure shows the detailed rules. The 47 mined rules specify the correlations between design parameter value pairs and the required process elements. In each rule, the support and the strength levels are also given. The number in the parentheses following the support value indicates the number of cases that satisfy the rule (in this case, the number of bracket “Ba” variants). For example, there are 3 bracket “Ba” variants that satisfy both the head and the body of rule #37: “*Number of fusing holes=3→ToolingI=LocatorAP with support 0.2 and strengthI*”.

Figure 7.1 Association rule mining in Magnum Opus

Figure 7.2 Identified association rules for the part family in Table 7.2

Further generalizing specific product related and process related items in rules in Figure 7.2 into their types, generic mapping rules are created for the mapping associations between bracket a family and its process element families. Some examples of generated generic rules are given in Figure 7.3 along with their occurring conditions and occurrence frequencies.

Generic Association Rules:

Rule 1: Generic bracket a $\Rightarrow$ Bracket a locator	Occurrence Frequency: 0.106
Occurring condition: If the thickness is between 1.26mm and 1.37mm, then the bracket a locator is used and the model is LocatorAP	
Rule 2: Generic bracket a $\Rightarrow$ Bracket a adjustor	Occurrence Frequency: 0.106
Occurring condition: If the number of fusing holes is 5, then the bracket a adjustor is used and the model is AdjustorDE.	
Rule 3: Generic bracket a $\Rightarrow$ Injection machine	Occurrence Frequency: 0.106
Occurring condition: If the material is ABS, then the injection machine is used and the model is InjectorHS35.	
Rule 4: Generic bracket a $\Rightarrow$ Bracket a locator	Occurrence Frequency: 0.063
Occurring condition: If the thickness is larger than 1.37, then the bracket a locator is used and the model is LocatorIII.	

Figure 7.3 Generalized generic mapping rules

7.5. Sensitivity Analysis

To evaluate the performance of the proposed association rule mining approach for mining associations between product and process variety in process platforms, sensitivity analysis of the mined association rules is performed. The results of association rule mining depend upon the specification of two thresholds: min_sup and min_conf . If the support and confidence levels are assigned with low values, many trivial rules will be generated. If the values of the two thresholds are specified too high, some useful patterns may be screened out.

Association rules basically suggest the mapping relationships between products and corresponding routings, more specifically variety parameter value pairs and process elements. Based on these rules, the proper process elements of the routing for a given product can be specified. The selection of process elements has a significant influence on the cycle time. As proved by Tieleman (1995), there is a positive relationship between costs and cycle time, that is, production costs increase with the cycle time. In this regard, this research introduces a performance measure for association rules, as follows:

$$\Psi(min_Sup, min_conf) = \alpha \sum_{i=1}^k T_i, \quad (7.6)$$

where α is the average cost per standard time unit, and T_i is the estimated cycle time for producing the i -th product of the family, which is deduced using the approach in Jiao and

Tseng (1999) based on specified process elements in rules discovered corresponding to given min_sup and min_conf .

Figure 7.4 Sensitivity analysis of association rules regarding support and confidence levels

The family of “bracket a” in the case study is used to conduct the sensitivity analysis. A total number of $19 \times 19 = 361$ runs of mining association rules are performed by enumerating all combinations of the min_sup and min_conf values, where both are changed from 0.05 to 0.95 with an increment of 0.05. In each run, routings for the 15 “bracket a” variants as shown in Table 7.2 are specified based on identified rules and their cycle times are computed as well. Cycle times $\{T_i\}_k$ are calculated as an average of the individual expected cycle times for the 15 variants. The final result is obtained and shown in Figure 7.4, where the performance measure is presented as a normalized comparison. The result of sensitivity analysis suggests that the optimal criteria of association rule mining are given as the support and confidence thresholds of 0.2 and 0.2, respectively.

7.6. Summary

In this chapter, variety coordination in process platforms is addressed through identifying mapping relationships between product and process variety. Taking advantage of the ability and flexibility of association rule mining and the availability of enormous

historical product and process data, an approach based on association rule mining is proposed to identify these mapping relationships. Compared against the traditional association rule mining, which tackles the associations between items from one domain, items in this research are from two domains: the product and process domains. The mined specific and generalized generic rules reflect the mapping relationships between product and process variety at different levels of abstraction and are used to determine process elements for given product items. In other words, they are conducive to specifying configuration rules in process platform-based production configuration. In the approach, mining association rules for parts and assemblies are dealt with differently, because the types of product-related items in their respective transaction databases are different. The reported case study of vibration motors proves the feasibility of the approach since the rules between product and process data in motor families are discovered quickly. Sensitivity analysis is conducted to evaluate the performance of the association rule mining approach by using different threshold values when mining the association rules. In the next chapter, a set of formalism will be introduced to shed light on the process of configuring routings from a process platform.

CHAPTER 8:

PRODUCTION CONFIGURATION MODELING USING NESTED COLORED OBJECT-ORIENTED PETRI NETS WITH CHANGEABLE STRUCTURES

In this chapter, production configuration is discussed from several perspectives, while the focus is on the process of configuring routings from a process platform. The proposed solution in this dissertation is to develop suitable modeling formalism and subsequently to model production configuration in terms of the configuration process. In relation to the problem nature and the principles of production configuration, the fundamental issues include variety handling, routing change accommodation, configuration at different levels of abstraction, and constraint satisfaction. To meet the modeling requirements posed by the fundamental issues, a set of formalism based on nested colored object-oriented PNs with changeable structures is developed in this research. A number of nets are defined to represent manufacturing processes, assembly process, parts, assemblies, conceptual processes of end products, and the internal behavior of manufacturing resources. Modeling production configuration using the developed formalism is demonstrated by the application case of vibration motors. In the next section, related work regarding PNs for systems modeling is presented.

8.1. Petri Nets for Systems Modeling

PN theory has been accepted as one of the first concurrent formalism to help solving real-time problems due to the ability to model, simulate and control complex flows and processes (Peterson, 1977; Peterson, 1981). PNs were first introduced by Petri in the early 1960s (Petri, 1962). Since then, they have been received intensive research and investigation attempting to tailor basic PN techniques to different problem domains and reduce the

modeling complexity. Among the many extensions obtained from these efforts, the colored PN (CPN, Jensen, 1992) and the object-oriented PN (OPN, Wang, 1996a; 1996b; 1996c) are two of the more popular ones. In CPN models of manufacturing systems, different colors, each of which indicates all the necessary data and information associated with objects, are assigned to tokens residing in places of the PN models. Thus, a concise, flexible and manageable representation of large manufacturing systems is facilitated (Jensen, 1992). Combining advantages of object-oriented techniques, e.g., the encapsulation of physical objects, the OPNs are developed to increase the maintainability and reusability of objects in modeling. Consequently, OPNs are powerful enough to model very large and complex systems or processes (Wang, 1996a; 1996b; 1996c). A further development based on the two concepts is the introduction of a mechanism that is intended to handle the structure changes of the system or process being modeled, i.e., PNs with changeable structures (Jiang et al., 1999). Morandin and Kato (2003) classified the PN extensions into two groups: the first deals with new attributes improving representation power, and the second deals with the new elements for systems representation. In spite of the advantages of PNs for system modeling and control, problems associated with PNs have been observed as well. As pointed out by Moore and Gupta (1996), the use of PNs for complex systems, or systems with large numbers of components, would produce models with a great number of elements in the graph, making it difficult to model and analyze the system's behavior.

PNs have been applied to a wide variety of problem domains, and their applications have been reported in the literature. A hierarchical PN framework was proposed for the representation of assembly plans and lower level control plans attempting to capture the hierarchical nature of assemblies in (Thomas et al., 1996). In the framework, tasks in a control plan corresponding to assembly robot operations are viewed as a lower level representation of tasks at the assembly plan level. A similar work was reported in (Adamou et al., 1998). The group introduced a hierarchical modeling of flexible assembly system control

with OPNs. The modeling includes a hierarchical decomposition of the system and coordination among lower-level decomposed system elements. The formalism of a three level scheme of PNs, where tokens can be PNs, was proposed for accommodating the modeling of multiple mobile robot system by Lopez-Mellado and Almeyda-Canepa (2003). The concept of virtual PN (VPN) was introduced to realize a modular modeling of the large and complex automated manufacturing systems (Morandin and Kato, 2003). In addition to the set of normal places and transitions, the virtual elements (i.e., virtual places and virtual transitions) in VPNs are the base for the connection of separated PN modules. Li and Lara-Rosano (1999) enhanced the object-oriented colored PNs (OOCPNs) with time delay and firing speed to a hybrid-like OOCPN intended to address the formal modeling of electronic component manufacturing systems, where parts are not processed one by one, but by batch. To ease the automatic generation of PN models for manufacturing system control and scheduling, He et al., (2000) proposed a decomposition methodology. In their methodology, IDEF3 is used to represent a manufacturing process first; then, the manufacturing process (i.e., the IDEF3 model) is decomposed into sub IDEF3 models based on similarity of resources. Subsequently, these decomposed IDEF3 models are transformed into corresponding PN control models. Finally, the sub-PN models are aggregated into a final complete one.

In addition to modeling manufacturing systems, PNs have been used in software system modeling. For example, Hiraishi (2000) presented a PN-based model adopting the classic place/transition nets to mathematically analyze and design multi agent systems. He represented agents using tokens that were place/transition nets also. The developed nested colored object-oriented PNs with changeable structures have adopted the same principle as that of the PN model of Hiraishi, wherein tokens in the higher level nets are detailed PNs at lower levels.

8.2. Production Configuration

This research defines production configuration as: *From an existing process platform in relation with a product family, the proper process elements, such as conceptual processes for individual product items (including the end products), operations, machines, tools, fixtures, cycle times, and setups are selected for new members of the associated product family, and subsequently arranged into routings, wherein the process concepts for items are replaced with the detailed operations, for producing the given products; both the selection and arrangement of process elements are subject to constraints represented by configuration rules in the process platform.*

Rather than dealing with product design, production configuration (i) takes the results of design (including the BOM and a list of product specifications) as input, (ii) configures the existing process elements from the process platform, and (iii) outputs an optimal routing that can produce the given product at the lower production cost and the shorter lead time. By optimal, it means producing the product using this routing will incur the fewest changes to the existing production line setups. In other words, the configured routing is the most similar to the existing ones on the shop floor, and therefore, the production variations are reduced to the minimum.

8.2.1. Production Configuration Process

As mentioned, the inputs to production configuration are detailed product specifications and/or BOM list. Further, the hierarchical structure of a product can be regarded as a collection of independent components organized at various levels of abstraction (Du et al., 2002b). Therefore, the process of production configuration is de facto an iterate process of configuring routings for each product item specified in the BOM. At each configuration, only the child items at the immediate lower level are considered.

Production configuration starts with the end product at the top level of the hierarchy. The results are (i) several conceptual processes for the immediate child items, (ii) assembly operations that are used to assemble the child items and the semi-items (formed by the child items), (iii) manufacturing resources that are required to complete the corresponding operations, the estimated cycle times for each operation and the corresponding setups, and (iv) the precedence between the operations. By conceptual processes, it means only the process concepts are selected for the child items rather than the process details, which are further elaborated in the following process configuration for the child items. Figure 8.1 illustrates the routing configuration for an item: *PI*, which has five immediate child items: A, B, C, D, and E. According to the input specifications, five conceptual processes for the five child items are selected first; the assembly operations joining the child items are also specified and ordered into a certain sequence.

Figure 8.1 Configuring process elements in production configuration

After configuring routings for the end product, production configuration proceeds to the child items at the immediate lower level. Each child item in turn is treated as an end product, and its conceptual process specified previously are refined accordingly. If the item being considered is an assembly, the elaborated process includes the conceptual processes of its immediate child items, assembly operations, manufacturing resources, estimated cycle times and setups associated with each assembly operation, and operations precedence. If the item is

a part, the refined process is a detailed manufacturing process. Such a detailed process consists of a number of sequenced manufacturing operations, each of which is attached with required manufacturing resources, an estimated cycle time and setups. After finishing configuring processes for all the items at the same level, production configuration recursively moves to the next lower level until the parts at the lowest level of the hierarchy are reached. The complete routing for the product is formed by replacing the conceptual processes at higher levels with the more refined processes at lower levels. During each configuration, the rules that establish the mapping relationships between product and process data as discussed in Chapter 7 in the process platform guide the selection of process elements and their connections as well. Figure 8.2 schematically shows production configuration for a product with an N level hierarchy.

Figure 8.2 Production configuration for a product with an N level hierarchy

As shown in the figure, the conceptual process for the end product corresponds to the items at the first level of the product hierarchy. Each of the process concepts in accordance with the relevant items is refined by specifying the conceptual processes for its child items at the second level of the product hierarchy, operations assembling these items, required

manufacturing resources, estimated cycle times, and setups for each operation, and operations precedence. In turn, the process concepts for the items at the second level are refined by more detailed processes concerning items at the third level. Therefore, for a product with an N -level hierarchy, the iterative process refinement will form an $N+1$ level process hierarchy, wherein the processes at the $(N+1)$ -th level are the detailed manufacturing processes for parts at the N -th level of the product hierarchy. Since parts can be at each level of the product hierarchy, detailed manufacturing processes thus can be found at each level of the $N+1$ process hierarchy except the first one.

8.2.2. Fundamental Issues

Configuring routings from a process platform is characterized by selection of proper process elements among existing ones and connecting the selected ones into complete routings while satisfying a number of constraints (configuration rules). In this process, more attention should be paid to the following fundamental issues.

(1) *Variety handling.* Individual customer expectations lead to a large number of product items, such as diverse parts and assemblies. These product items are specified by various design parameters in conjunction with particular value instances. On the shop floor, producing such a variety of parts, assemblies and final products necessitates distinct processes, operations and manufacturing resources. It is imperative to deal with the large number of product and process elements while leveraging upon existing product families and process platforms.

(2) *Process change.* In spite of the similarity inherent in product and process families, each product is fulfilled by using a unique production process. Production processes of different products may differ from one another in specific operations, manufacturing resources, and process flows. In addition, within a company's manufacturing capabilities, there may exist a number of alternative production processes for producing one product. Such

a wide range of possible changes in production processes must be explicitly considered in order to determine an optimal production configuration.

(3) *Levels of abstraction.* Prasad (1998) studies the product and process complexity associated with providing variety and highlights the importance of determining the right amount of decomposition. This leads to a granularity paradox, indicating tradeoffs associated with process details and product hierarchies. Production configuration must support refinement of decision-making through decomposition at different levels of abstraction.

(4) *Constraint satisfaction.* In general, production configuration involves four types of constraints. Inclusion conditions specify the circumstances under which a process concept and an operation are to be included in a configuration. This type of constraint coincides with the fact that not all of the processes and operations in a process platform are necessary for producing each individual product. The sequence or precedence relations among operations processes constitute another type of constraints. Each sequence constraint is specified as a binary relationship between two processes or operations in the form of a predecessor and a successor. The third type of constraint is commonly used to determine execution rules regarding operation details, for example, to specify machines and tools to be used for a particular product. Finally, coordination constraints must be addressed in order to maintain the consistency of product and process variety. More specifically, these constraints control the granularity of variety derivation from certain product hierarchies to proper process details.

8.2.3. Configuration Description

The following definitions provide the problem nature of production configuration.

Definition 8-1: A process platform, Ω , consisting of a set of production process variants, $\{R_r\}_N$, for producing the set of product variants in a family, is defined as a tuple, $\Omega = \langle \Psi, \rightarrow \rangle$, where $\Psi = \{P_i\}_n$ is a set of process classes, each of which is for producing a

family of product items, may it be a part type or an assembly type; and $\rightarrow$ is the sequence relation between two process classes in Ψ .

Definition 8-2: The sequence relation $\rightarrow$ defines a binary relationship between two process classes, P_i and P_j , $\forall i \neq j \in [1, n]$, such that $P_i \rightarrow P_j$, indicating that P_i should be completed before commencement of P_j . Furthermore, a transitive closure of $\rightarrow$ is reflexive, so that with Ψ the set of $\rightarrow$ forms a tree.

Definition 8-3: With respect to various product item families, Ψ can be further classified into two sets, i.e., $\Psi = \Psi^M \cup \Psi^S$, where $\Psi^M = \{P_i^M\}_a$ is a set of master process classes that are compulsory to all process variants; $\Psi^S = \{P_i^S\}_b$ is set of selective process classes that are optional to process variants; and $a + b = n$.

If $P_i \in \Psi^M$, P_i is said to produce a family of assemblies or parts that are common to all variants in the product family. If $P_i \in \Psi^S$, it means that assemblies or parts produced by any instance of P_i are optional to the set of product variants.

Definition 8-4: Each P_i is defined as a tuple, $P_i = \langle \omega_i, \succ \rangle$, where $\omega_i = \{O_{ij}\}_m$ is a set of operations classes, and $\succ$ is the precedence relation between two operations classes in ω_i .

Definition 8-5: The precedence relation $\succ$ defines a binary relationship between two operations classes, O_{is} and O_{it} , $\forall s \neq t$, such that $O_{is} \succ O_{it}$, meaning that O_{is} should be performed before O_{it} . A transitive closure of $\succ$ is reflexive, so that with ω_i the set of $\succ$ forms a tree representing P_i .

Definition 8-6: Two sets are distinguished within ω_i of P_i , i.e., $\omega_i = \omega_i^M \cup \omega_i^S$, where $\omega_i^M = \{O_{ij}^M\}_x$ is a set of master operation classes necessary to all $\{R_r\}_N$; $\omega_i^S = \{O_{ij}^S\}_y$ is a set of selective operation classes optional to $\{R_r\}_N$; and $x + y = m$. For each O_{ij} of P_i , if $O_{ij} \in \omega_i^M$, then $P_i \in \Psi^M$. Likewise, if $P_i \in \Psi^S$, then either $O_{ij} \in \omega_i^M$ or $O_{ij} \in \omega_i^S$.

Definition 8-7: A production process variant, R_r , consists of a series of sequenced processes, i.e., $R_r = \langle \Psi_r^*, \rightarrow \rangle$, where $\Psi_r^* = \{P_{ri}^*\}_n$ is a set of specific processes, each of which is to produce a particular product item, may it be a part or an assembly, and $\rightarrow$ is the sequence relation between two processes, P_{ra}^* and P_{rb}^* , $\forall a \neq b$, such that $P_{ra}^* \rightarrow P_{rb}^*$, suggesting that process P_{ra}^* for producing item Ia must be completed before producing item Ib through P_{rb}^* .

Definition 8-8: Each P_{ri}^* of production process variant R_r contains an operations set, i.e., $P_{ri}^* = \langle \omega_{ri}^*, \succ \rangle$, where $\omega_{ri}^* = \{O_{rij}^*\}_m$ is a set of specific operations, and $\succ$ is the precedence relation between operations in $\{O_{rij}^*\}_{N \times n \times m}$, such that $O_{ris}^* \succ O_{rit}^*$, $\forall s \neq t$, denoting that O_{ris}^* must be performed before O_{rit}^* .

A process class P_i belongs to one of three types: (1) a type of manufacturing processes consisting of a series of machining operations and non-machining operations for manufacturing a part family, e.g., material transfer; (2) a type of assembly processes involving a series of assembly operations and non-assembly operations for producing an assembly family; or (3) a mixed type of processes involving machining operations, assembly operations, and/or non-machining/assembly operations for forming an assembly family. If P_{ri}^* is to produce a part, then its corresponding $\{O_{rij}^*\}_{N \times n \times m}$ comprises all machining operations. If P_{ri}^* is to form an assembly, then O_{rij}^* may be a machining operation or an assembly operation.

Both manufacturing and assembly processes employ a number of material handling devices, a number of buffers, and a set of machine classes. Each machine class in turn exhibits a number of similar machines. These machines are either necessary or optional for producing a product family. Material handlers such as AGVs, robots, and human operators may be used to transfer materials, semi-finished items (i.e., Work in Process (WIP)), and

finished items from one location to another. A number of buffers, including input buffers, WIP buffers, and output buffers, are also used to store materials, WIP and finished items or products, respectively. Figure 8.3 summarizes the process elements and relationships involved in production configuration.

Figure 8.3 Elements and their connections in production configuration

8.3. Modeling Formalism

8.3.1. Overview

The fundamental issues in production configuration pose a number of unique requirements on the modeling formalism as listed next.

- (1) To accommodate the involved large variety in the way that a concise and easy understandable model can be built;
- (2) To handle routing structure changes so that the built model can be adapted without any difficulties to different configurations;
- (3) To address issues concerning the concept selection first, and then granular refinement of these selected concepts till all details have been worked out; and
- (4) To deal with multiple constraints at different granularity.

In order to capture all the fundamental issues, the set of modeling formalism based on nested colored object-oriented PNs with changeable structures has been developed. The principles of CPNs, OPNs and the mechanism for handling structure changes in PN models are adopted to define the nets in the formalism. The relevant data regarding product item, process elements and manufacturing resources are attached to colored tokens in CPNs to tackle multiple constraints. Together with OPNs, they deal with the large and various variety involved. The change handling mechanism is intended to address the modeling of routing structure changes. Moreover, the concept of net nesting is introduced for coping with granularity issues such that lower level nets that represent more detailed processes are nested in places of higher level nets as colored tokens. In the modeling formalism, four different kinds of PNs are defined. A resource net (*RNet*) is specified to reflect the internal behaviors of physical objects (i.e., the set of manufacturing resources). A manufacturing net (*MNet*) is defined to reflect both the manufacturing process of parts and parts themselves when it is nested in a place of a higher level net. An assembly net (*ANet*) is introduced to represent the process of producing assemblies. Similar to the *MNet*, it is used to denote the produced assemblies when it is nested in a place of the higher level PN. The process net (*PNet*) is used to describe the conceptual process of the end product. While in a system model of production configuration constructed by using the developed modeling formalism there may exist a number of *MNets*, *ANets*, and *RNets*, only one *PNet* can be found.

8.3.2. Net Definitions

8.3.2.1. Resource Net

A production system comprises a set of input and output buffers, WIP buffers, machining and assembly machines, as well as material handlers. Each of them performs certain functionality that is achieved through the internal activities, and thus they can be

modeled as OPNs from an object-oriented perspective. Their internal activities are reflected in the associated OPNs (Wang, 1996a).

Definition 8-9: A resource net is defined as a tuple, $RNet = (G^{OPN}, \omega)$, where $RNet$ is a resource net, may it be an input buffer, output buffer, WIP buffer, machining machine, assembly machine, or material handler. G^{OPN} is a structure of OPN, such that,

$$G^{OPN} = (SP, IM, OM, AT, C^{SP}, C^{IM}, C^{OM}, C^{AT}, I^{id}, O^{id}),$$

where $SP = \{sp_i\}_{N^{SP}}$ is a finite set of state places; $IM = \{im_k\}_{N^{IM}}$ is a finite set of input message places, $SP \cap IM = \emptyset$, $SP \cup IM \neq \emptyset$; $OM = \{om_l\}_{N^{OM}}$ is a finite set of output message places, $IM \cap OM = \emptyset$, $SP \cap OM = \emptyset$, $SP \cap IM \cap OM = \emptyset$; and $AT = \{at_j\}_{N^{AT}}$ is a finite set of activity transitions.

$C^{SP}(i) = \{c_j^{sp_i}\}_{N^{sp_i}}$ is a set of colored tokens associated with a state place, sp_i .

$C^{IM}(k) = \{c_j^{im_k}\}_{N^{im_k}}$ is a set of colored tokens associated with an input message place, im_k .

$C^{OM}(l) = \{c_j^{om_l}\}_{N^{om_l}}$ is a set of colored tokens associated with an output message place, om_l .

$C^{AT}(j) = \{c_i^{at_j}\}_{N^{at_j}}$ is a set of colored tokens associated with an activity transition, at_j .

Moreover, $C^{SP}(i)$, $C^{IM}(k)$, $C^{OM}(l)$, and $C^{AT}(j)$ are determined by the items that the resource objects can process.

$$I^{id}(SP, AT/c^{at_j}): C^{SP} \times C^{AT} \rightarrow C^{SP} \cup \emptyset \text{ or } I^{id}(IM, AT/c^{at_j}): C^{IM} \times C^{AT} \rightarrow C^{IM} \cup \emptyset$$

is an input identity function for arcs that connect state places SP or input message places IM to activity transitions AT with a firing colored token, $c^{at_j} \in C^{AT}(j)$, $\exists j \in [1, N^{AT}]$.

$$O^{id}(SP, AT/c^{at_j}): C^{SP} \times C^{AT} \rightarrow C^{SP} \cup \emptyset \text{ or } O^{id}(OM, AT/c^{at_j}): C^{OM} \times C^{AT} \rightarrow C^{OM} \cup \emptyset$$

is an output identity function for arcs that connect activity transitions AT with a firing colored token, c^{at_j} , to state places, SP , or output message places, OM .

$\omega : (SP \rightarrow M^{C^{SP}}, IM \rightarrow M^{C^{IM}}, OM \rightarrow M^{C^{OM}})$ is a marking function, such that $\omega(SP) = M^{C^{SP}}, \omega(IM) = M^{C^{IM}}, \omega(OM) = M^{C^{OM}}$, where $M^{C^{SP}}, M^{C^{IM}}$ and $M^{C^{OM}}$ are the families of all multisets over C^{SP}, C^{IM} and C^{OM} , respectively.

The set of state places, $\{sp_i\}_{N^{SP}}$, specify all possible states that a resource object may possess. As shown in Figure 8.4 and Table 8.1, a machine has three states, namely idle, setting up and processing. A material handler possesses three states as well, i.e., idle, transferring parts from buffers to machines, and transferring parts from machines to buffers. All types of buffers assume two states: idle/capacity available and loading parts. The **AND** relations among input arcs indicate that tokens in input places must appear at the same time to enable the activity transitions with certain colored tokens. Similarly, after transition firing colored tokens should be simultaneously added into places with **AND** relation arcs.

Figure 8.4 States and activities of resource objects

To enable communication among objects, input message places, IM , and output message places, OM , are defined as the interface for objects to send and receive messages (i.e., tokens). For example, in Figure 8.4, im_i of the *RNet* for a material handler is used for receiving those messages sent by machines with request for loading parts. om_i holds the messages to be sent to other machines about the completion of part transferring. *AT* describes all activities that a resource object can perform. Input identity functions

$I^{id}(SP, AT/c^{at_j})$ and $I^{id}(IM, AT/c^{at_j})$ specify the set of pre-conditions for enabling AT with a colored token, c^{at_j} , whilst output identity functions $O^{id}(SP, AT/c^{at_j})$ and $O^{id}(OM, AT/c^{at_j})$ regulate the set of post-conditions of the firing of AT . ω represents the marking of the resource object after each transition firing.

Table 8.1 Legend in Figure 8.4

Place	Material Handler (MH) RNet	Machine RNet	Buffer RNet
im_1	A request (from machine) for transferring part in	A request (from buffer) for setting up	A message (from MH) of parts ready to be moved in
im_2	A request (from machine) for transferring part out	A request (from MH) for loading parts	
om_1	A message of part transferring completion	A message of completing setup	A message of parts ready to be moved out
om_2	A message of part transferring completion	A message of completing part unloading	
sp_1	Transferring parts into machine	Setting up	Part loaded in buffer
sp_2	Idle	Idle	Idle and capacity available
sp_3	Transferring parts out of machine	Processing parts	
at_1	Start transferring parts into machine	Start setting up	Loading part into buffer
at_2	End transferring parts	End setting up	Unloading part from buffer
at_3	Start transferring parts out of machine	Start loading parts	
at_4	End transferring parts out	End part unloading	

The dynamic behavior of a resource object is characterized by the set of state places, $\{sp_i\}_{N^{SP}}$, and the set of activity transitions, $\{at_j\}_{N^{AT}}$. Only when the connected input state place sp_i and message place im_k hold the colored tokens as specified by the input identity functions, i.e., $\omega(sp_i) \geq I^{id}(sp_i, at_j/c^{at_j})$ and $\omega(im_k) \geq I^{id}(im_k, at_j/c^{at_j})$, can an activity of at_j be carried out, that is, activity transition at_j is activated. The firing of at_j results in the removal of tokens from the set of input places as specified by the input identity functions and the addition of colored tokens to the set of connected output places as specified in the output identify functions.

8.3.2.2. Manufacturing Net

Definition 8-10: A manufacturing net is defined as a tuple, $MNet = (G^{OPNs-cs}, \mu)$, where $MNet$ is a manufacturing net representing the processes of manufacturing a part family. $G^{OPNs-cs}$ is a system of OPNs with changeable structures, such that,

$$G^{OPNs-cs} = (P, T, C^P, C^T, I, O),$$

where $P = \{p_i\}_{N^P}$ is a finite set of places, representing the set of resource objects involved in the manufacturing process; $T = \{t_j\}_{N^T}$ is a finite set of transitions, $P \cup T \neq \emptyset$, $P \cap T = \emptyset$, $T = T_1 \cup T_2$, where T_1 is a set of input *OR* relation transition (i.e., transitions with input arcs bearing *OR* relations), and T_2 is a set of ordinary transitions; $C^P(i) = \{c_j^{p_i}\}_{N^{p_i}}$ is a finite set of colored tokens associated with a place, p_i ; and $C^T(j) = \{c_i^{t_j}\}_{N^{t_j}}$ is a finite set of colored tokens associated with a transition, t_j .

$I(p_i, t_j / c^{t_j}): C^P(i) \times C^T(j) \rightarrow C^P(i) \cup \emptyset$ is an input transform function for an arc, (p_i, t_j) , $\forall p_i \in {}^{\circ}t_j$, that connects a place to a transition with a firing colored token, c^{t_j} . Likewise, $O(p_i, t_j / c^{t_j}): C^P(i) \times C^T(j) \rightarrow C^P(i) \cup \emptyset$ is an output transform function for an arc, (p_i, t_j) , $\forall p_i \in t_j^{\circ}$, that connects a transition with a firing colored token, c^{t_j} , to a place. Moreover, $\mu: P \rightarrow M^{C^P}$ is a marking function, $\mu(P) = M^{C^P}$, where M^{C^P} is the family of all multisets over C^P .

To accommodate process variations, e.g., adding or removing manufacturing resources, or changing the execution order of two machining or non-machining operations, the OPNs with changeable structures (OPNs-cs) in (Jiang et al., 1999) is adopted to define the net structure, $G^{OPNs-cs}$, of an *MNet*. The set of places, $\{p_i\}_{N^P}$, are defined for the set of resource objects. Each place p_i carries out either a machining operation or a non-machining operation. Therefore, the set of colored tokens, $\{c_j^{p_i}\}_{N^{p_i}}$, associated with each place p_i are used to represent the operations, the employed resource objects, as well as the corresponding product items.

Furthermore, the set of transitions $\{t_j\}_{N^T}$ specify the starting or ending of the relevant operations. The colored token of each transition t_j is defined exactly the same as that of its input place. The input and output transform functions specify the type and number of colored tokens to be removed/added from/to the relevant input and output places. In other words, each transition t_j , colored token set $\{c_i^{t_j}\}_{N^{t_j}}$, and the associated input and output transform functions determine the message passing relationship between two places, p_a and p_b , in accordance with the flow of a part's manufacturing process. If a transition, t_j , is to be activated with respect to a firing colored token, c^{t_j} , each input place p_i must contain colored tokens, the type and number of which should be greater or equal to the one specified by the input transform function, i.e., $\mu(p_i) \geq I(p_i, t_j / c^{t_j})$, $p_i \in {}^\circ t_j$.

8.3.2.3. Assembly Net

Definition 8-11: An assembly net is defined as a tuple, $ANet = (G^{OPNs-cs}, \mu)$, where $ANet$ is an assembly net representing the assembly processes of an assembly family. $G^{OPNs-cs}$ is a system of OPNs with changeable structures, $G^{OPNs-cs} = (P, T, C^P, C^T, I, O)$, where T , C^T , I , O , and μ carry the same meaning as that for $MNet$.

P is a finite set of places, $\{p_i\}_{N^P}$, $P = P^1 \cup P^2$, where $P^1 = \{p_i^1\}_{N^{P^1}}$ is a set of places corresponding to $MNets$ and/or lower level $ANets$ of child assemblies; $P^2 = \{p_i^2\}_{N^{P^2}}$ is a set of places representing the resource objects and the performed subassembly processes; and $P^1 \cap P^2 = \emptyset$, $N^{P^1} + N^{P^2} = N^P$.

C^P is a finite set of colored tokens associated with all places, $C^P = C^{P^1} \cup C^{P^2}$, where each $C^{P^1}(i) = \{c_j^{p_i^1}\}_{N^{P_i^1}}$ is a set of colored tokens associated with a place, p_i^1 , indicating the set of part or assembly variants of a family produced by the nested $MNet$ or $ANet$ at the

lower level; and $C^{P^2}(j) = \{c_i^{p_j^2}\}_{N^{p_j^2}}$ is a set of colored tokens associated with a resource object p_j^2 .

An *ANet* is defined to represent the processes of producing a family of assemblies. Changes to component items that form assembly variants cause variations in the corresponding assembly processes. To deal with such process changes, the structural change handling mechanism is introduced to the *ANet*. Unlike those places in an *MNet*, the places in an *ANet* may not be always related to the resource objects, but rather, they are also defined for the child parts and child assemblies produced by the *MNets* or *ANets*. If a place, p_i , is defined for *MNets* or *ANets* at a lower level, the set of colored tokens, $\{c_j^{p_i^l}\}_{N^{IFp_i^l}}$, is specified to indicate the set of similar item variants (either parts or assemblies), which are produced by the processes nested in the places. If it represents a resource object, the set of colored tokens is assigned according to the process flow of the assembly and used to indicate the assembly operations or non-assembly operations along with the associated resource objects.

8.3.2.4. Process Net

Definition 8-12: A process net is defined as a tuple, $PNet = (G^{OPNs-cs}, \mu)$, where $PNet$ is a process net representing the conceptual production processes of producing a family of end products. $G^{OPNs-cs}$ is a system of OPNs with changeable structures, such that $G^{OPNs-cs} = (P, T, C^P, C^T, I, O)$, where T , C^T , I , O , and μ carry the same meaning as that for an *MNet*.

$P = \{p_i\}_{N^P}$ is a finite set of places, $P = P^M \cup P^A \cup P^R$, where $P^M = \{p_i^M\}_{N^{p^M}}$ is a set of places corresponding to *MNets*; $P^A = \{p_i^A\}_{N^{p^A}}$ is a set of places representing *ANets*;

$P^R = \{p_i^R\}_{N^{P^R}}$ is a set of places indicating resource objects and operations performed; and

$$P^M \cap P^A \cap P^R = \phi, N^{P^M} + N^{P^A} + N^{P^R} = N^P.$$

C^P is a finite set of colored tokens associated with all places, $C^P = C^{P^M} \cup C^{P^A} \cup C^{P^R}$, where each $C^{P^M}(i) = \{c_j^{P_i^M}\}_{N^{PFp_i^M}}$ is a set of colored tokens associated with a place, p_i^M , and indicates the set of similar part variants produced by the represented *MNet*; $C^{P^A}(i) = \{c_j^{P_i^A}\}_{N^{AFp_i^A}}$ is a set of colored tokens associated with a place, p_i^A , and indicates the set of assembly variants produced by the represented *ANet*; and $C^{P^R}(i) = \{c_j^{P_i^R}\}_{N^{P_i^R}}$ is the set of colored tokens associated with the set of resource objects along with the performed operations.

A *PNet* is defined for the abstract processes of producing a family of end products. Similar to an *ANet*, the structural change handling mechanism is employed in *PNet* to handle process variations. The places in a *PNet* are specified to represent either resource objects carrying out certain operations, or *MNets* manufacturing parts, or *ANets* producing assemblies. If a place, p_i , is defined for an *MNet* or *ANet*, then the set of colored tokens, $\{c_j^{P_i^M}\}_{N^{PFp_i^M}}$ or $\{c_j^{P_i^A}\}_{N^{AFp_i^A}}$, is specified to indicate the set of part or assembly variants in a family. If a place, p_i , is specified to represent a resource object, the set of colored tokens is used to indicate the operations carried out by the objects.

To specify firing conditions for transitions with respect to firing colored tokens, each of these transitions, i.e., T_l in an *MNet*, *ANet* or *PNet*, is decomposed into several input transitions, along with a state place and an output transition. As shown in Figure 8.5, the decomposed input and output transform functions as well as the state place are assigned with the same colored tokens as those associated with the input places of the original transition.

Figure 8.5 Transition decomposition

For a single resource object, whose number is one, there may exist more than one input arc. Therefore, conflict may occur when multiple objects or subprocesses require a single object to perform multiple operations at the same time. To maintain a 1-bounded property and the safeness of an object place, inhibitor arcs, $Inh(p_i, t_j), \forall t_j \in {}^o p_i$, are introduced to these objects (Peterson, 1977; Wang and Wu, 1998). The inhibitor arcs of a resource object are drawn from the set of input transitions to the single object. Different from those general input arcs, inhibitor arcs are indicated as dotted line with circles at the transition ends. If $Inh(p_i, t_j) = 1$, it implies that no operation request can be passed to the object represented by the place p_i unless the object is not occupied, that is, there is no token in the place.

8.3.2.5. Nested Net System

Definition 8-13: A multilevel nested net system is defined as a tuple, $NNSys = (PNet, R, M, A, PP, SP, \varphi)$, where $NNSys$ is the multilevel nested net system for modeling production configuration; $PNet$ is the process net describing conceptual production processes of a product family; and $\varphi(pp_i): PP \rightarrow SP$ is a port place assignment function. It is defined from PP to SP , so as to establish a binary relationship between PP and SP .

$R = \{r_i\}_{N^R}$ is a finite set of $RNets$, $R = R^1 \cup R^2 \cdots \cup R^O$, where each $R^o = \{r_i^o\}_{N^{R^o}}, \forall o \in [1, O]$, is a finite set of $RNets$ representing the behaviors of the objects

that are in the same *MNets* and *ANets*, or the *PNet*. $M = \{m_i\}_{N^M}$ is a finite set of *MNets*, $M = M^1 \cup M^2 \dots \cup M^P$, where each $M^p = \{m_i^p\}_{N^{M^p}}$, $\forall p \in [1, P]$, is a finite set of *MNets* nested in such places that are in the same nets at the immediate higher levels. Likewise, $A = \{a_i\}_{N^A}$ is a finite set of *ANets*, $A = A^1 \cup A^2 \dots \cup A^Q$, where each $A^q = \{a_i^q\}_{N^{A^q}}$, $\forall q \in [1, Q]$ represents the set of *ANets* that are nested in the same nets at the immediate higher levels.

$PP = \{pp_i\}_{N^{PP}}$ is a finite set of port places representing resource objects in A and M .

The messages in the output message places of each pp_i are sent to the input message places of such resource objects that are connected with those places representing *MNets* or *ANets*, to which these port places belong. In other words, the places representing message receiving objects and the places that nest *MNets* or *ANets* of port places belong to the same nets.

$SP = \{sp_j\}_{N^{SP}}$ is a finite set of socket places indicating resource objects in A and the *PNet*. The input message places of each sp_j receives messages sent from resource objects of the associated pp_i that belong to the lower level *MNets* or *ANets*, which are represented by the places connected to sp_i in the same *ANets* or the *PNet*.

$\varphi(pp_i): PP \rightarrow SP$ is a port place assignment function; it is defined from PP into SP so that a binary relation exists between PP and SP .

Performing as an abstraction mechanism, *NNSys* facilitates the selection of conceptual processes with right amount of details. Within *NNSys*, the highest level is the *PNet*, while a number of *RNets*, *MNets* and *ANets* are located at the second level. Each of these nets provides more detail for the respective places in the *PNet*. The nets at any lower level provide detailed descriptions of the assembly and manufacturing processes nested in the places of the nets at its immediate higher level. At the lowest level of each path, all nets become *RNets*, whilst a mixture of *RNets*, *MNets* and *ANets* can be found at any arbitrary

level. Figure 8.6 demonstrates an $N+2$ level net system with nested $MNets$ and $ANets$, as well as encapsulated $RNets$. Table 8.2 summarizes descriptions of places and transitions of different nets at each level. The set of activity transitions, state places, input/output message places of $RNets$ are explained in Table 8.1.

Figure 8.6 System model for production configuration modeling

Table 8.2 Legends for Figure 8.6

Place	Level 1	Level 2	Level N+1
p_1	Part a_i & its manufacturing process	Part $a_i p_i$ of A_i & its manufacturing process	Input buffer in a part's <i>MNet</i>
p_2	Assembly A_i & its assembly process	Part $a_i p_2$ of A_i & its manufacturing process	A machining machine & its machining operation
p_3	An assembly machine & its operation	An assembly machine & its operation in A_i 's <i>ANet</i>	A material handler in a part's <i>MNet</i>
p_4	A material handler & its operation	A material handler & its operation in A_i 's <i>ANet</i>	WIP buffer in a part's <i>MNet</i>
p_5	Output buffer in the final	WIP buffer in A_i 's <i>ANet</i>	A machining machine & its machining operation
p_6		Part $a_i p_3$ of A_i & its manufacturing process	A machining machine & its machining operation
p_7		An assembly machine & its operation in A_i 's <i>ANet</i>	A machining machine & its machining operation
p_8		Output buffer in A_i 's <i>ANet</i>	Output buffer of a part's <i>MNet</i>
dp_1		Decomposed state place of transition t_1 in A_i 's	Decomposed state place of transition t_1 in
dp_2		Decomposed state place of transition t_2 in A_i 's	Decomposed state place of transition t_2 in
dp_4		Decomposed state place of transition t_4 in A_i 's	Decomposed state place of transition t_4 in
$dt_{11/12/13}$		Decomposed transitions of t_1 in A_i 's <i>ANet</i>	Decomposed transitions of t_1 in the <i>MNet</i>
$dt_{21/22/23/24/25}$		Decomposed transitions of t_2 in A_i 's <i>ANet</i>	Decomposed transitions of t_2 in the <i>MNet</i>
t_3	A transition in the <i>PNet</i>	A transition in A_i 's <i>ANet</i>	A transition in a part's <i>MNet</i>
$dt_{41/42/43/44/45}$		Decomposed transitions of t_4 in A_i 's <i>ANet</i>	Decomposed transitions of t_4 in the <i>MNet</i>
t_5	A transition in the <i>PNet</i>	A transition in A_i 's <i>ANet</i>	A transition in a part's <i>MNet</i>

To specify firing conditions, transitions that have *OR* relation input arcs have been decomposed into state places, input and output transitions as shown. In addition, the set of inhibitor arcs drawn as dashed lines are applied to the set of places representing such single manufacturing resources that are shared by several resources (or operations).

To enable the communication through sending and receiving messages between objects at two adjacent levels, the port places, *PP*, in the lower level nested nets and socket places, *SP*, in the higher level nets are introduced and defined for resource objects only. The specification of port and socket places attempts to address the connection between lower and higher level nets and thus the continuity of the modeling from the lowest level to the highest level. For example, as shown in Figure 8.7 (two levels in a *NNSys*), when a token representing a part is produced in the *MNet* at level $i+1$, which is nested in place p_2 in the *ANet* at level i , and loaded into the output buffer represented by place p_6 , a token with the same color appears in place p_2 in the *ANet* at level i . Meanwhile, a message requesting machine setup from the output buffer p_6 in the nested *MNet* is sent to the place p_3 representing an assembly machine in the *ANet* at level i .

Figure 8.7 Communication between port and socket places

8.3.3. Production Configuration Modeling

Consistent with the hierarchical structure of a product, production configuration can be regarded as a recursive process of configuring process elements through various levels of abstraction. For a product variant with an N level hierarchy, production configuration is carried out for various product items at each level. Figure 8.8 draws an analogy of decision making among the product hierarchy, production configuration and the nested net system model. For a product with an N -level hierarchy, the iterative process refinement will form a process hierarchy with $N+1$ levels, wherein the processes at the $(N+1)$ -th level are the detailed manufacturing processes for parts at the N -th level of the product hierarchy. Since parts can be at each level of the product hierarchy, detailed manufacturing processes can thus be found at each level of the $N+1$ process hierarchy except the first level. In relation to the detailed manufacturing processes at the $(N+1)$ -th level of the process hierarchy, *MNets* are constructed at level $N+1$ of the nested net system model. *RNets* representing internal behaviors of resource objects in the *MNets* at level $N+1$ are established at level $N+2$ of the production configuration model. Therefore, the production configuration model for a product with an N level product hierarchy will assume $N+2$ levels.

Figure 8.8 Analogy of decision making for granularity

8.4. Case Study

⋮

8.4.1 Production Configuration Modeling

To test the developed modeling formalism and present the process of production configuration the same vibration motors from the previous case studies are used. Figure 8.9 shows the hierarchies of two motor variants, VM_1 and VM_2 . The nodes in the figure represent the set of specific parts and assemblies that compose VM_1 and VM_2 . Table 8.3 lists all the descriptions of these items.

The motor process platform has been constructed a priori, including sets of master and selective processes and various process elements regarding operations, precedence, manufacturing resources, and their produced product items. In addition, execution rules for operation details, inclusion conditions for selective process elements and other constraints regulating execution orders of processes and operations have been identified by domain experts.

Figure 8.9 Product hierarchies of VM_1 and VM_2 Table 8.3 Specific items of VM_1 and VM_2

Abbreviations	Items
VM_1/VM_2	Vibration motor variants 1 and 2
Wt_1/Wt_2	Weight variants 1 and 2
Rh_1/Rh_2	Rubber holder variants 1 and 2
MB_1/MB_2	Mainbody variants 1 and 2
FA_1/FA_2	Frame assy variants 1 and 2
AA_1/AA_2	Armature assy variants 1 and 2
BA_1/BA_2	Bracket assy variants 1 and 2
CA_1/CA_2	Coil assy variants 1 and 2
Fm_1/Fm_2	Frame variants 1 and 2
Mt_1	Magnet variant 1
St_1/St_2	Shaft variants 1 and 2
Ba_1/Ba_2	Bracket a variants 1 and 2
Bb_1/Bb_2	Bracket b variants 1 and 2
Tl_1/Tl_2	Terminal variants 1 and 2
Cl_1/Cl_2	Coil variants 1 and 2
Tp_1/Tp_2	Tape variants 1 and 2

In accordance with the specifications of the three child items at the first level of the product hierarchies, the two conceptual processes are determined for VM_1 and VM_2 , respectively. A conceptual production process reflects the assembly flow of a number of items (parts and/or assemblies) rather than the detailed processes of manufacturing a part. Both processes require Wcaulking machine (WcM_1), whereas the production of VM_1 needs another assembly machine RhI_1 (rubber holder inserter 1). The process flow for VM_1 is as follows. Part Wt_1 and assembly MB_1 are produced $\rightarrow$ the assembly machine WcM_1 sets up for the two material items (i.e., Wt_1 and MB_1) $\rightarrow$ Wt_1 and MB_1 are transferred by a material handler to WcM_1 $\rightarrow$ WcM_1 assembles Wt_1 and MB_1 thus forming a semi-finished item $\rightarrow$ the

formed semi-finished item is transferred by the material handler from WcM_1 to the WIP buffer $\rightarrow$ the semi-finished item is loaded into WIP buffer and another part Rh_1 is produced $\rightarrow RhI_1$ sets up for the semi-finished product and $Rh_1 \rightarrow$ the semi-finished item and Rh_1 are transferred by material handler to $RhI_1 \rightarrow$ the assembly machine RhI_1 assembles the two items and produces the end product $VM_1 \rightarrow VM_1$ is transferred by the material handler from RhI_1 to the output buffer. In the process of VM_2 , there is only one assembly operation performed by WcM_1 .

The process flow of VM_2 follows next. Part Wt_2 and assembly MB_2 are produced $\rightarrow WcM_1$ sets up for the two material items $\rightarrow Wt_2$ and MB_2 are transferred by the material handler to $WcM_1 \rightarrow WcM_1$ joins Wt_2 and MB_2 and forms the end motor variant $VM_2 \rightarrow VM_2$ is transferred by the material handler from WcM_1 to the output buffer. Table 8.4 summarizes the above mentioned two flows of VM_1 and VM_2 along with required manufacturing resources. Also given in the table are places representing operations and resource objects in the built *PNet* for the two end products along with colored tokens assigned to them.

Table 8.4 Process flows of VM_1 and VM_2

Process flow of VM_1			Process flow of VM_2		
Operations	Resource(or process) /Place	Colored Token	Operations	Resource(or process) /Place	Colored Token
Part Wt_1 is produced	Manufacturing process/ p_1	Wt_1	Part Wt_2 is produced	Manufacturing	Wt_2
Assembly MB_1 is produced	Assembly process/ p_2	MB_1	Assembly MB_2 is produced	Assembly process/ p_2	MB_2
WcM_1 setting up	WcM_1 / p_3	$VM_1 \cdot 0$	WcM_1 setting up	WcM_1 / p_3	$VM_2 \cdot 0$
Being transferred	Material handler/ p_4	$VM_1 \cdot 1$	Being transferred	Material handler/ p_4	$VM_2 \cdot 1$
Being assembled	WcM_1 / p_3	$VM_1 \cdot 2$	Being assembled	WcM_1 / p_3	$VM_2 \cdot 2$
Being transferred	Material handler/ p_4	$VM_1 \cdot 3$	Being transferred	Material handler/ p_4	$VM_2 \cdot 3$
Staying in buffer	WIP buffer/ p_5	$VM_1 \cdot 4$	Staying in buffer	Output buffer/ p_8	VM_2
Part Rh_1 is produced	Manufacturing process/ p_6	Rh_1			
RhI_1 setting up	RhI_1 / p_7	$VM_1 \cdot 5$			
Being transferred	Material handler/ p_4	$VM_1 \cdot 6$			
Being assembled	RhI_1 / p_7	$VM_1 \cdot 7$			
Being transferred	Material handler/ p_4	$VM_1 \cdot 8$			
Staying in buffer	Output buffer/ p_8	VM_1			

The $PNet$ is then constructed according to the process flows of the configured conceptual processes, as shown in Figure 8.10. In the figure, dt_{xx} and dp_{xx} denote the decomposed transitions and state places of three transitions, T_1, T_2 and T_3 , of which the input arcs bear OR relations. In addition, the inhibitor arcs have been applied to three single resources, including a material handler (p_4) and two assembly machines, WcM_1 (p_3) and RhI_1 (p_7). The colored token in place p_8 (i.e., the output buffer) indicates that one VM_1 or VM_2 has been produced. To deal with the process change, i.e., removal of RhI_1 from VM_1 's process, the structure change handling mechanism modifies the input/output transfer functions, i.e., message sending and receiving relations in the $PNet$.

Figure 8.10 The $PNet$ of VM_1 and VM_2

Figure 8.11 The $MNet$ for part variants Wt_1 and Wt_2

Table 8.5 gives the detailed list of colored tokens assigned to places and transitions (including the decomposed transitions and state places), as well as the input and output transform functions. Generalizing the two variant case to the whole motor family, the colored tokens assigned to places p_1 , p_2 and p_6 indicate all the family members that are produced

by the nested processes. The tokens assigned to place p_8 (the output buffer) thus represent the entire motor family.

Table 8.5 Colored tokens, input and output transform functions for the $PNet$

Place	Colored Token	Transition	Colored Token	Input Arc	Input Arc Transform Function	Output Arc	Output Arc Transform Function
p_1	$\{Wt_1, Wt_2\}$	dt_{11}	$\{Wt_1MB_1, Wt_2MB_2\}$	(p_1, dt_{11})	$\{Wt_1MB_1, Wt_2MB_2\}$	(dp_1, dt_{11})	$\{Wt_1MB_1, Wt_2MB_2\}$
p_2	$\{MB_1, MB_2\}$			(p_2, dt_{11})	$\{Wt_1MB_1, Wt_2MB_2\}$		
p_3	$\{VM_1 \cdot 0, VM_1 \cdot 2, VM_2 \cdot 0, VM_2 \cdot 2\}$	dt_{12}	$\{VM_1 \cdot 4Rh_1\}$	(p_3, dt_{12})	$\{VM_1 \cdot 4Rh_1\}$	(dp_1, dt_{12})	$\{VM_1 \cdot 4Rh_1\}$
p_4	$\{VM_1 \cdot 1, VM_1 \cdot 3, VM_2 \cdot 1, VM_2 \cdot 3, VM_1 \cdot 6, VM_1 \cdot 8\}$	dt_{13}	$\{VM_1 \cdot 4Rh_1, Wt_1MB_1, Wt_2MB_2\}$	(dp_1, dt_{13})	$\{VM_1 \cdot 4Rh_1, Wt_1MB_1, Wt_2MB_2\}$	(p_3, dt_{13})	$\{VM_1 \cdot 0, VM_2 \cdot 0\}$
						(p_7, dt_{13})	$\{VM_1 \cdot 5\}$
p_5	$\{VM_1 \cdot 4\}$	dt_{21}	$\{VM_1 \cdot 0, VM_2 \cdot 0\}$	(p_5, dt_{21})	$\{VM_1 \cdot 0, VM_2 \cdot 0\}$	(dp_2, dt_{21})	$\{VM_1 \cdot 0, VM_2 \cdot 0\}$
p_6	$\{Rh_1\}$	dt_{22}	$\{VM_1 \cdot 5\}$	(p_7, dt_{22})	$\{VM_1 \cdot 5\}$	(dp_2, dt_{22})	$\{VM_1 \cdot 5\}$
p_7	$\{VM_1 \cdot 5, VM_1 \cdot 7\}$	dt_{23}	$\{VM_2 \cdot 0, VM_1 \cdot 0, VM_1 \cdot 5\}$	(dp_2, dt_{23})	$\{VM_2 \cdot 0, VM_1 \cdot 0, VM_1 \cdot 5\}$	(p_4, dt_{23})	$\{VM_2 \cdot 1, VM_1 \cdot 1, VM_1 \cdot 6\}$
p_8	$\{VM_1, VM_2\}$	t_3	$\{VM_2 \cdot 1, VM_1 \cdot 1, VM_1 \cdot 6\}$	(p_4, t_3)	$\{VM_2 \cdot 1, VM_1 \cdot 1, VM_1 \cdot 6\}$	(p_3, t_3)	$\{VM_1 \cdot 2, VM_2 \cdot 2\}$
dp_1	$\{VM_1 \cdot 4Rh_1, Wt_1MB_1, Wt_2MB_2\}$					(p_7, t_3)	$\{VM_1 \cdot 7\}$
dp_2	$\{VM_2 \cdot 0, VM_1 \cdot 0, VM_1 \cdot 5\}$	dt_{41}	$\{VM_1 \cdot 2, VM_2 \cdot 2\}$	(p_3, dt_{41})	$\{VM_1 \cdot 2, VM_2 \cdot 2\}$	(dp_4, dt_{41})	$\{VM_1 \cdot 2, VM_2 \cdot 2\}$
		dt_{42}	$\{VM_1 \cdot 7\}$	(p_7, dt_{42})	$\{VM_1 \cdot 7\}$	(dp_4, dt_{42})	$\{VM_1 \cdot 7\}$
dp_4	$\{VM_2 \cdot 2, VM_1 \cdot 2, VM_1 \cdot 7\}$	dt_{43}	$\{VM_2 \cdot 2, VM_1 \cdot 2, VM_1 \cdot 7\}$	(dp_4, dt_{43})	$\{VM_2 \cdot 2, VM_1 \cdot 2, VM_1 \cdot 7\}$	(p_4, dt_{43})	$\{VM_2 \cdot 3, VM_1 \cdot 3, VM_1 \cdot 8\}$
		t_5	$\{VM_2 \cdot 3, VM_1 \cdot 3, VM_1 \cdot 8\}$	(p_4, t_5)	$\{VM_2 \cdot 3, VM_1 \cdot 3, VM_1 \cdot 8\}$	(p_5, t_5)	$\{VM_1 \cdot 4\}$
						(p_8, t_5)	$\{VM_1 \cdot 2, VM_2 \cdot 2\}$

Figure 8.11 illustrates the $MNet$ for a part family, consisting of two variants, Wt_1 and Wt_2 . Their respective manufacturing processes are configured, as shown in Table 8.6, where the colored tokens are specified to represent both the operations and the employed manufacturing resources. While both of them employ the same machine (WMM_1 and WMM_2), the manufacturing processes differ from each other in the process flow. The places associated with the net are defined as shown in Table 8.7. Table 8.8 summarizes specifications of all the places, transitions, colored tokens, input and output arcs and transform functions defined for the $MNet$. The colored tokens are assigned to different places, thus differentiating specific operations related to individual part variants of the family.

Table 8.6 Process flows of part variants Wt_1 and Wt_2

Process flow of Wt_1			Process flow of Wt_2		
Operations	Resource/Place	Colored Token	Operations	Resource/Place	Colored Token
Staying in buffer	Input buffer/ p_1	$Wt_1 \cdot 0$	Staying in buffer	Input buffer/ p_1	$Wt_2 \cdot 0$
WMM_1 setting up	WMM_1 / p_2	$Wt_1 \cdot 1$	WMM_2 setting up	WMM_2 / p_3	$Wt_2 \cdot 1$
Being transferred	Material handler/ p_3	$Wt_1 \cdot 2$	Being transferred	Material handler/ p_3	$Wt_2 \cdot 2$
Being machined	WMM_1 / p_2	$Wt_1 \cdot 3$	Being machined	WMM_2 / p_3	$Wt_2 \cdot 3$
Being transferred	Material handler/ p_3	$Wt_1 \cdot 4$	Being transferred	Material handler/ p_3	$Wt_2 \cdot 4$
Staying in buffer	WIP buffer/ p_4	$Wt_1 \cdot 5$	Staying in buffer	WIP buffer/ p_4	$Wt_2 \cdot 5$
WMM_2 setting up	WMM_2 / p_5	$Wt_1 \cdot 6$	WMM_1 setting up	WMM_1 / p_2	$Wt_2 \cdot 6$
Being transferred	Material handler/ p_3	$Wt_1 \cdot 7$	Being transferred	Material handler/ p_3	$Wt_2 \cdot 7$
Being machined	WMM_2 / p_5	$Wt_1 \cdot 8$	Being machined	WMM_1 / p_2	$Wt_2 \cdot 8$
Being transferred	Material handler/ p_3	$Wt_1 \cdot 9$	Being transferred	Material handler/ p_3	$Wt_2 \cdot 9$
Staying in buffer	Output buffer/ p_6	Wt_1	Staying in buffer	Output buffer/ p_6	Wt_2

Table 8.7 Resources and operations associated with the $MNet$ and the $ANet$

Place	$ANet$ of assemblies MB_1 & MB_2	$MNet$ of part Rh_1	$MNet$ of parts Wt_1 & Wt_2
p_1	Assembly processes of assemblies AA_1 & AA_2	Input buffer	Input buffer
p_2	Assembly processes of assemblies BA_1 & BA_2	Machining machine $RhSM_1$	Machining machine WMM_1
p_3	Assembly machine AIM_1	Material handler	Material handler
p_4	Material handler	WIP buffer	WIP buffer
p_5	WIP buffer	Machining machine $RhSM_2$	Machining machine WMM_2
p_6	Assembly process of assembly FA_1	Machining machine $RhSM_3$	Output buffer
p_7	Assembly machine FCM_1	Machining machine $RhSM_4$	
p_8	Output buffer	Output buffer	

Table 8.8 Colored tokens, input and output transform functions for the $MNet$

Place	Colored Token	Transition	Colored Token	Input Arc	Input Arc Transform Function	Output Arc	Output Arc Transform Function
p_1	$\{Wt_1 \cdot 0, Wt_2 \cdot 0\}$	dt_{11}	$\{Wt_1 \cdot 0, Wt_2 \cdot 0\}$	(p_1, dt_{11})	$\{Wt_1 \cdot 0, Wt_2 \cdot 0\}$	(dp_1, dt_{11})	$\{Wt_1 \cdot 0, Wt_2 \cdot 0\}$
p_2	$\{Wt_1 \cdot 1, Wt_1 \cdot 3, Wt_2 \cdot 6, Wt_2 \cdot 8\}$	dt_{12}	$\{Wt_1 \cdot 5, Wt_2 \cdot 5\}$	(p_4, dt_{12})	$\{Wt_1 \cdot 5, Wt_2 \cdot 5\}$	(dp_1, dt_{12})	$\{Wt_1 \cdot 5, Wt_2 \cdot 5\}$
		dt_{13}	$\{Wt_1 \cdot 0, Wt_1 \cdot 5, Wt_2 \cdot 0, Wt_2 \cdot 5\}$	(dp_1, dt_{13})	$\{Wt_1 \cdot 0, Wt_1 \cdot 5, Wt_2 \cdot 0, Wt_2 \cdot 5\}$	(p_2, dt_{13}) (p_5, dt_{13})	$\{Wt_1 \cdot 1, Wt_2 \cdot 6\}$ $\{Wt_1 \cdot 6, Wt_2 \cdot 1\}$
p_3	$\{Wt_1 \cdot 2, Wt_1 \cdot 4, Wt_1 \cdot 7, Wt_1 \cdot 9, Wt_2 \cdot 2, Wt_2 \cdot 4, Wt_2 \cdot 7, Wt_2 \cdot 9\}$	dt_{21}	$\{Wt_1 \cdot 1, Wt_2 \cdot 6\}$	(p_2, dt_{21})	$\{Wt_1 \cdot 1, Wt_2 \cdot 6\}$	(dp_2, dt_{21})	$\{Wt_1 \cdot 1, Wt_2 \cdot 6\}$
		dt_{22}	$\{Wt_1 \cdot 6, Wt_2 \cdot 1\}$	(p_3, dt_{22})	$\{Wt_1 \cdot 6, Wt_2 \cdot 1\}$	(dp_2, dt_{22})	$\{Wt_1 \cdot 6, Wt_2 \cdot 1\}$
p_4	$\{Wt_1 \cdot 5, Wt_2 \cdot 5\}$	dt_{23}	$\{Wt_1 \cdot 1, Wt_1 \cdot 6, Wt_2 \cdot 1, Wt_2 \cdot 6\}$	(dp_2, dt_{23})	$\{Wt_1 \cdot 1, Wt_1 \cdot 6, Wt_2 \cdot 1, Wt_2 \cdot 6\}$	(p_3, dt_{23})	$\{Wt_1 \cdot 2, Wt_1 \cdot 7, Wt_2 \cdot 2, Wt_2 \cdot 7\}$
p_5	$\{Wt_1 \cdot 6, Wt_1 \cdot 8, Wt_2 \cdot 1, Wt_2 \cdot 3\}$	t_3	$\{Wt_1 \cdot 2, Wt_1 \cdot 7, Wt_2 \cdot 2, Wt_2 \cdot 7\}$	(p_3, t_3)	$\{Wt_1 \cdot 2, Wt_1 \cdot 7, Wt_2 \cdot 2, Wt_2 \cdot 7\}$	(p_2, t_3) (p_5, t_3)	$\{Wt_1 \cdot 3, Wt_2 \cdot 8\}$ $\{Wt_1 \cdot 8, Wt_2 \cdot 3\}$
p_6	$\{Wt_1, Wt_2\}$	dt_{41}	$\{Wt_1 \cdot 3, Wt_2 \cdot 8\}$	(p_2, dt_{41})	$\{Wt_1 \cdot 3, Wt_2 \cdot 8\}$	(dp_4, dt_{41})	$\{Wt_1 \cdot 3, Wt_2 \cdot 8\}$
dp_1	$\{Wt_1 \cdot 0, Wt_1 \cdot 5, Wt_2 \cdot 0, Wt_2 \cdot 5\}$	dt_{42}	$\{Wt_1 \cdot 8, Wt_2 \cdot 3\}$	(p_3, dt_{42})	$\{Wt_1 \cdot 8, Wt_2 \cdot 3\}$	(dp_4, dt_{42})	$\{Wt_1 \cdot 8, Wt_2 \cdot 3\}$
dp_2	$\{Wt_1 \cdot 1, Wt_1 \cdot 6, Wt_2 \cdot 1, Wt_2 \cdot 6\}$	dt_{43}	$\{Wt_1 \cdot 3, Wt_1 \cdot 8, Wt_2 \cdot 3, Wt_2 \cdot 8\}$	(dp_4, dt_{43})	$\{Wt_1 \cdot 3, Wt_1 \cdot 8, Wt_2 \cdot 3, Wt_2 \cdot 8\}$	(p_3, dt_{43})	$\{Wt_1 \cdot 4, Wt_1 \cdot 9, Wt_2 \cdot 4, Wt_2 \cdot 9\}$
dp_4	$\{Wt_1 \cdot 3, Wt_1 \cdot 8, Wt_2 \cdot 3, Wt_2 \cdot 8\}$	t_3	$\{Wt_1 \cdot 4, Wt_1 \cdot 9, Wt_2 \cdot 4, Wt_2 \cdot 9\}$	(p_3, t_3)	$\{Wt_1 \cdot 4, Wt_1 \cdot 9, Wt_2 \cdot 4, Wt_2 \cdot 9\}$	(p_4, t_3) (p_6, t_3)	$\{Wt_1 \cdot 5, Wt_2 \cdot 5\}$ $\{Wt_1, Wt_2\}$

As shown in Figure 8.12, the *ANet* for assembly variants MB_1 and MB_2 is constructed according to the configured conceptual process flows in Table 8.9. The places in relation to the respective manufacturing resources and the performed operations are shown in Table 8.7. The mechanism for handling process changes is adopted in the *ANet* to describe that fact that assembly machine FCM_1 is required for producing MB_1 , but not for MB_2 . Table 8.10 summarizes specifications of all the places, transitions, colored tokens, input and output arcs and transform functions defined for the *ANet*. Places p_1 , p_2 and p_6 represent the respective assembly processes of three families: AAs, BAs and FAs. The tokens are specified for each of the places, denoting specific assembly variants of a family.

Figure 8.12 The *ANet* for assembly variants MB_1 and MB_2 Table 8.9 Process flows of assembly variants MB_1 and MB_2

Process flow of MB_1			Process flow of MB_2		
Sub Process	Resource(or process)/Place	Colored Token	Sub Process	Resource(or process)/Place	Colored Token
Assembly AA_1 is produced	Assembly process/ p_1	AA_1	Assembly AA_2 is produced	Assembly process/ p_1	AA_2
Assembly BA_1 is produced	Assembly process/ p_2	BA_1	Assembly BA_2 is produced	Assembly process/ p_2	BA_2
AIM_1 setting up	$AIM_1 \mid p_3$	$MB_1 \cdot 0$	AIM_1 setting up	$AIM_1 \mid p_3$	$MB_2 \cdot 0$
Being transferred	Material handler/ p_4	$MB_1 \cdot 1$	Being transferred	Material handler/ p_4	$MB_2 \cdot 1$
Being assembled	$AIM_1 \mid p_3$	$MB_1 \cdot 2$	Being assembled	$AIM_1 \mid p_3$	$MB_2 \cdot 2$
Being transferred	Material handler/ p_4	$MB_1 \cdot 3$	Being transferred	Material handler/ p_4	$MB_2 \cdot 3$
Staying in buffer	WIP buffer/ p_5	$MB_1 \cdot 4$	Staying in buffer	Output buffer/ p_8	MB_2
Assembly FA_1 is produced	Assembly process/ p_6	FA_1			
FCM_1 setting up	$FCM_1 \mid p_7$	$MB_1 \cdot 5$			
Being transferred	Material handler/ p_4	$MB_1 \cdot 6$			
Being assembled	$FCM_1 \mid p_7$	$MB_1 \cdot 7$			
Being transferred	Material handler/ p_4	$MB_1 \cdot 8$			
Staying in buffer	Output buffer/ p_8	MB_1			

Table 8.10 Colored tokens, input and output transform functions for the *ANet*

Place	Colored Token	Transition	Colored Token	Input Arc	Input Arc Transform Function	Output Arc	Output Arc Transform Function
p_1	$\{AA_1, AA_2\}$	dt_{11}	$\{AA_1BA_1, AA_2BA_2\}$	(p_1, dt_{11})	$\{AA_1BA_1, AA_2BA_2\}$	(dp_1, dt_{11})	$\{AA_1BA_1, AA_2BA_2\}$
p_2	$\{BA_1, BA_2\}$			(p_2, dt_{11})	$\{AA_1BA_1, AA_2BA_2\}$		
p_3	$\{MB_1 \cdot 0, MB_2 \cdot 2\}$	dt_{12}	$\{MB_1 \cdot 4FA_1\}$	(p_3, dt_{12})	$\{MB_1 \cdot 4FA_1\}$	(dp_1, dt_{12})	$\{MB_1 \cdot 4FA_1\}$
p_4	$\{MB_1 \cdot 1, MB_2 \cdot 3, MB_2 \cdot 1, MB_2 \cdot 3, MB_1 \cdot 6, MB_1 \cdot 8\}$	dt_{13}	$\{MB_1 \cdot 4FA_1, AA_1BA_1, AA_2BA_2\}$	(dp_1, dt_{13})	$\{MB_1 \cdot 4FA_1, AA_1BA_1, AA_2BA_2\}$	(p_3, dt_{13})	$\{MB_1 \cdot 0, MB_2 \cdot 0\}$
						(p_7, dt_{13})	$\{MB_1 \cdot 5\}$
p_5	$\{MB_1 \cdot 4\}$	dt_{21}	$\{MB_1 \cdot 0, MB_2 \cdot 0\}$	(p_3, dt_{21})	$\{MB_1 \cdot 0, MB_2 \cdot 0\}$	(dp_2, dt_{21})	$\{MB_1 \cdot 0, MB_2 \cdot 0\}$
p_6	$\{FA_1\}$	dt_{22}	$\{MB_1 \cdot 5\}$	(p_7, dt_{22})	$\{MB_1 \cdot 5\}$	(dp_2, dt_{22})	$\{MB_1 \cdot 5\}$
p_7	$\{MB_1 \cdot 5, MB_1 \cdot 7\}$	dt_{23}	$\{MB_1 \cdot 0, MB_2 \cdot 0, MB_1 \cdot 5\}$	(dp_2, dt_{23})	$\{MB_1 \cdot 0, MB_2 \cdot 0, MB_1 \cdot 5\}$	(p_4, dt_{23})	$\{MB_1 \cdot 0, MB_2 \cdot 0, MB_1 \cdot 5\}$
p_8	$\{MB_1, MB_2\}$	t_3	$\{MB_1 \cdot 1, MB_2 \cdot 1, MB_1 \cdot 6\}$	(p_4, t_3)	$\{MB_1 \cdot 1, MB_2 \cdot 1, MB_1 \cdot 6\}$	(p_3, t_3)	$\{MB_1 \cdot 2, MB_2 \cdot 2\}$
						(p_7, t_3)	$\{MB_1 \cdot 7\}$
dp_1	$\{MB_1 \cdot 4FA_1, AA_1BA_1, AA_2BA_2\}$	dt_{41}	$\{MB_1 \cdot 2, MB_2 \cdot 2\}$	(p_3, dt_{41})	$\{MB_1 \cdot 2, MB_2 \cdot 2\}$	(dp_4, dt_{41})	$\{MB_1 \cdot 2, MB_2 \cdot 2\}$
dp_2	$\{MB_2 \cdot 0, MB_1 \cdot 0, MB_1 \cdot 5\}$	dt_{42}	$\{MB_1 \cdot 7\}$	(p_7, dt_{42})	$\{MB_1 \cdot 7\}$	(dp_4, dt_{42})	$\{MB_1 \cdot 7\}$
		dt_{43}	$\{MB_1 \cdot 2, MB_2 \cdot 2, MB_1 \cdot 7\}$	(dp_4, dt_{43})	$\{MB_1 \cdot 2, MB_2 \cdot 2, MB_1 \cdot 7\}$	(p_4, dt_{43})	$\{MB_1 \cdot 2, MB_2 \cdot 2, MB_1 \cdot 7\}$
dp_4	$\{MB_1 \cdot 2, MB_1 \cdot 7\}$	t_5	$\{MB_1 \cdot 3, MB_2 \cdot 3, MB_1 \cdot 8\}$	(p_4, t_5)	$\{MB_1 \cdot 3, MB_2 \cdot 3, MB_1 \cdot 8\}$	(p_3, t_5)	$\{MB_1 \cdot 4\}$
						(p_8, t_5)	$\{MB_1, MB_2\}$

The *MNet*, *ANet* and *PNet* are constructed to model the manufacturing and assembly processes of parts, assemblies and final products, respectively. The internal behavior of each resource object is modeled by constructing an *RNet*. Figure 8.13 shows an example of the *RNet* for the final product assembly machine WcM_1 in the *PNet* (Figure 8.10).

Figure 8.13 Internal behaviors of the assembly machine WcM_1

In accordance with the product hierarchies in Figure 8.9, a number of *RNets*, *MNets* and *ANets* are constructed for all the parts, assemblies and manufacturing resources contained in the process platform. Then the system model for production configuration is composed based on net nesting, as shown in Figure 8.14. For illustrative simplicity, not all

the *RNets*, *MNets* and *ANets* are shown in the figure. Also not included is level 6 for the *RNets*.

Figure 8.14 Nested net system model of configuring routings for VM_1 and VM_2

Upon completion of producing a particular item, the objects represented by port places start to send messages that request for certain operations to the objects represented by socket places. For example, when one MB_2 is produced in the *ANet* at level 2 and loaded into output buffer p_8 (also a port place), a message requesting machine setup from p_8 is sent to

socket place p_3 (representing machine WcM_1) within the $PNet$. Meanwhile, place p_2 (nesting the $ANet$) holds the same colored token as that of MB_2 . The inhibitor arcs are then applied to those single resource objects, for example, machine WcM_1 (p_3) and the material handler (p_4) within the $PNet$. As a consequence, each time objects are guaranteed to perform one task only.

8.4.2. System Analysis

Deadlock detection and conflict prevention are widely adopted as performance indicators for testing a built system model (Wang, 1996; Wang and Wu, 1998). This section adopts these criteria for performance evaluation of the nested net system model. The focus is on the $MNet$ for Wt_1 and Wt_2 .

Generally there are two types of input conflict and one type of output conflict (Wang, 1996). Type I input conflicts involve more than one output transition to be connected to one output place. Type II conflicts involve two or more input places to be connected to one transition via an *OR* relation. Whenever a transition possesses more than one output place that is connected through an *OR* relation, an output conflict may occur.

Figure 8.15 The $MNet$ of part variants Wt_1 and Wt_2

Figure 8.15 shows the *MNet* for part family *Wt*. It contains no type II input conflicts, as all of them are removed by transition decomposition. A type I input conflict may occur at places p_2 and p_5 , since both of them are connected to two output transitions. An output conflict may occur at transitions dt_{13} , t_3 and t_5 , since all of them exhibit two output places with *OR* relations. Through the assignment of colored tokens to places and transitions and in conjunction with input and output transform functions, the decisions regarding which transition to fire and which place to add tokens to can be determined according to the colors. Therefore, output conflicts are resolved by assigning different colors.

The deadlock detection algorithm in Wang and Wu (1998) is applied to the *MNet* in Figure 8.15, as elaborated next.

(1) The initial state is set as that raw materials of two parts are in the input buffer, and the goal state is set as that both two parts are produced and loaded into the output buffer. The initial marking M_0 and the goal state marking M_g become the following,

$$\begin{aligned} M_0 &= (\mu(p_1), \mu(p_2), \mu(p_3), \mu(p_4), \mu(p_5), \mu(p_6), \mu(dp_1), \mu(dp_2), \mu(dp_4)) \\ &= (Wt_1 \cdot 0 + Wt_2 \cdot 0, 0, 0, 0, 0, 0, 0, 0, 0) \end{aligned}$$

$$M_g = (0, 0, 0, 0, 0, Wt_1 + Wt_2, 0, 0, 0).$$

(2) Construct an incidence matrix, W , such that,

$$W^+ = [w_{ij}^+]_{m \times n}, \text{ where } w_{ij}^+ = O(p_i, t_j / c^{t_j}), \forall i \in [1, m], j \in [1, n]; \text{ and}$$

$$W^- = [w_{ij}^-]_{m \times n}, \text{ where } w_{ij}^- = I(p_i, t_j / c^{t_j}), \forall i \in [1, m], j \in [1, n].$$

Set $m = 9$ (the total number of places in the *MNet*) and $n = 11$ (the total number of transitions in the *MNet*). Then,

$$\begin{aligned}
W^+ &= \begin{bmatrix} dt_{11} & dt_{12} & dt_{13} & dt_{21} & dt_{22} & dt_{23} & t_3 & dt_{41} & dt_{42} & dt_{43} & t_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & O_1 & 0 & 0 & 0 & O_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & O_3 & 0 & 0 & 0 & O_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & O_5 \\ 0 & 0 & O_6 & 0 & 0 & 0 & O_7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & O_8 \\ id & id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & id & id & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & id & id & 0 & 0 \end{bmatrix} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_6 \\ dp_1 \\ dp_2 \\ dp_3 \end{matrix} \\
W^- &= \begin{bmatrix} id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & id & 0 & 0 & 0 & id & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & id & 0 & 0 & 0 & id \\ 0 & id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & id & 0 & 0 & 0 & id & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & id & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & id & 0 & 0 \end{bmatrix} \\
W = W^+ - W^- &= \begin{bmatrix} -id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & O_1 & -id & 0 & 0 & O_2 & -id & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & O_3 & -id & 0 & 0 & O_4 & -id \\ 0 & -id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & O_5 \\ 0 & 0 & O_6 & 0 & -id & 0 & O_7 & 0 & -id & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & O_8 \\ id & id & -id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & id & id & -id & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & id & id & -id & 0 \end{bmatrix}.
\end{aligned}$$

(3) At the initial marking M_0 (i.e., $k=0$), the set of enabled transitions, $T_{enable-0}$, are related to their firing colored tokens c^{t_j} , that is, $T_{enable-0} = \{dt_{11}/Wt_1 \cdot 0, dt_{11}/Wt_2 \cdot 0\}$.

(4) Transition $dt_{11}/Wt_1 \cdot 0$ is selected to fire.

(5) The characteristic vector $\overline{S}_k$ of a firing sequence S is set to have entry c^{t_j} for the transition selected to fire and 0 for all others. Hence,

$$\begin{aligned}
\overline{S}_0 &= (dt_{11}, dt_{12}, dt_{13}, dt_{21}, dt_{22}, dt_{23}, t_3, dt_{41}, dt_{42}, dt_{43}, dt_5)^{-1} \\
&= (Wt_1 \cdot 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^{-1}.
\end{aligned}$$

(6) The following state marking, $M_{k+1} = M_{0+1} = M_1$, after firing the transition $dt_{11}/Wt_1 \cdot 0$ is computed according to $M_{k+1}^{-1} = M_k^{-1} + W\overline{S}_k$. Thereby, $M_1 = (Wt_2 \cdot 0, 0, 0, 0, 0, 0, 0, Wt_1 \cdot 0, 0, 0, 0)$.

As $M_1 \neq M_g$, k is set to be $k+1$ (i.e., $k=0+1=1$). Then go to step (3).

As for marking M_1 , it is true that $T_{enable-1} = \{dt_{13}/Wt_1 \cdot 0, dt_{11}/Wt_2 \cdot 0\}$. Thus transition $dt_{11}/Wt_2 \cdot 0$ is selected to fire. Then $\overline{S}_1 = (Wt_2 \cdot 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^{-1}$. Therefore

$M_{k+1} = M_2$ is computed as $M_2 = (0, 0, 0, 0, 0, 0, Wt_1 \cdot 0 + Wt_2 \cdot 0, 0, 0)$. Since $M_2 \neq M_g$, k is set to 2. Go to step (3).

Since $T_{enable-2} = \{dt_{13}/Wt_1 \cdot 0, dt_{13}/Wt_2 \cdot 0\}$, $dt_{13}/Wt_1 \cdot 0$ is selected to fire. As a result, $\overline{S_2} = (0, 0, Wt_1 \cdot 0, 0, 0, 0, 0, 0, 0)^{-1}$, and thus M_3 is computed as $M_3^{-1} = M_2^{-1} + W\overline{S_2}$. According to the output functions of the *MNet* (Figure 8.14), we can infer that $O_6(Wt_1 \cdot 0) = 0$ and $O_1(Wt_1 \cdot 0) = Wt_1 \cdot 1$. With the input identity function defined for the *MNet*, $-id(Wt_1 \cdot 0) = -Wt_1 \cdot 0$ is obtained. Furthermore,

$$M_3^{-1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ Wt_1 \cdot 0 + Wt_2 \cdot 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & O_1 & -id & 0 & 0 & O_2 & -id & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & O_3 & -id & 0 & 0 & O_4 \\ 0 & -id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & O_5 \\ 0 & 0 & O_6 & 0 & -id & 0 & O_7 & 0 & -id & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & O_8 \\ id & id & -id & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & id & id & -id & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & id & id & -id & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ Wt_1 \cdot 0 + Wt_2 \cdot 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ O_1(Wt_1 \cdot 0) \\ 0 \\ 0 \\ O_6(Wt_1 \cdot 0) \\ 0 \\ -id(Wt_1 \cdot 0) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ Wt_1 \cdot 0 + Wt_2 \cdot 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ Wt_1 \cdot 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -Wt_1 \cdot 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ Wt_2 \cdot 0 \\ 0 \\ 0 \end{bmatrix}$$

Since $M_3 \neq M_g$, the deadlock detection process is continued. Upon completion, the deadlock analysis reaches a final goal marking, $M_g = (0, 0, 0, 0, 0, 0, Wt_1 + Wt_2, 0, 0)$. Figure 8.16 shows an example of feasible sequence that leads to the final goal state. Accordingly, it concludes that the *MNet* for Wt_1 and Wt_2 is live and deadlock free.

Figure 8.15 Procedure of deadlock detection algorithm

Figure 8.16 A feasible firing sequences leading to the goal marking

8.5. Summary

This chapter addresses production configuration, more specifically the process of configuring routings from a process platform. To shed light on how routings are configured,

the proposed solution in this dissertation is to model production configuration. In production configuration, a large variety of product and process elements are involved; a number of configuration rules require to be satisfied for selecting and arranging process elements; process elements are selected for product items at different levels of a product hierarchy; and routings to be configured are different with one another. The modeling formalism must have the ability to address explicitly the afore mentioned fundamental issues in production configuration so as to build accurately system models. A set of modeling formalism based on nested colored object-oriented PNs with changeable structures is developed to conduct production configuration modeling. The modeling abilities of the set of formalism lie in colored tokens for dealing with constraints and variety handling, OPNs for handling variety, nested net concept for addressing configuration granularity concern, and the changeable structural mechanism for handling routing variations. The results of the application case not only clarify the process of production configuration but also show the correctness of the system model through deadlock and conflict analysis. In the next chapter, conclusions will be drawn and future work will be pointed out.

CHAPTER 9:

CONCLUSIONS AND FUTURE WORK

9.1. Summary of Research Work

The literature review reveals the common deficiencies and limitations of the related research, namely platform-based product family design, product configuration, production planning in mass customization, integrated product and process data management, and reconfigurable manufacturing systems. This dissertation puts forward process platform-based production configuration for companies to configure routings (for new members of a product family) by using the existing process elements and knowledge. In addition to supporting routing planning, process platforms provide well-structured mechanisms for companies to manage product and process variety effectively within a coherent framework.

Chapter 3 discusses process platform-based production configuration in general. Combining the principles of product configuration and routing planning, process platform-based production configuration is able to provide the most similar routings for producing individualized products through configuring existing process elements. Fundamental issues in process platform-based production configuration are identified in three areas: (1) process platform, (2) production configuration, and (3) configuration evaluation. In each area, this dissertation identifies several more specific research issues. In the process platform area, issues related to process platform concepts, functionalities, constituent elements, and construction as well as variety coordination in process platforms are discussed. In the production configuration area, issues are associated with production configuration formulation, configuration space modeling, and production configuration modeling. In the configuration evaluation area, issues are in relation to evaluation of process platforms to be developed and routings to be configured from the associated process platforms.

Chapter 4 defines mathematical models to formulate rigorously a process platform with respect to concept definitions and functionalities. OO concepts and set theory are adopted in this research in order to build process platform models accurately. The formulation is accomplished by establishing models of the generic structures basic to process platforms, data representation solutions and generic planning for production configuration. The variety coordination mechanism embedded in a process platform is formulated based on variety parameters and their possible value sets. To achieve this, the concepts of variety handlers and their special states are introduced for variety parameters and the value instances.

Process platforms are analyzed with respect to the constituent elements and their relationships in Chapter 5. In order to represent precisely various classes and their member instances as well as to capture the unique relationships among them, i.e., class-to-class, class-to-member, and member-to-member, UML models of process platforms are constructed. The structural representation of process platforms is approached from the generic product structures, the generic process structures, and the integrated generic routing structures with focus on correspondence between the generic product and process structures.

Since a generic routing structure underpins the associated process platform to be constructed, process platform construction can be facilitated by forming the generic routings of the corresponding process families. In Chapter 6, a systematic data mining methodology is introduced to identify generic routings for process families. Data mining techniques are employed due to the availability of the enormous product and process data in companies' databases and the presence of advanced data mining tools. For solving the unique mining problem, wherein both textual data and structural data are involved in routings, text mining and tree matching techniques are adopted in the methodology. Mining data to identify generic routings encompasses three sequential stages, including routing similarity measure, routing clustering and routing unification. In measuring routing similarity, the pairwise comparison

of given routings is accomplished by measuring node content similarity and structure similarity of routing representation trees. Subsequently, based on the result of pairwise comparison, i.e., a routing similarity matrix, the fuzzy clustering approach in the methodology distributes similar routings into same families. At last, routing unification takes a family of routings as input and outputs the generic routing of the input family.

Chapter 7 reports an association rule mining approach to discovering associations between product and process data in a process platform. These associations de facto reflect the mapping relationships between product and process variety, which is exhibited by the large amount of product and process data documented in a process platform. Variety coordination in process platforms is thus supported by identifying the mapping relationships between product and process variety.

In Chapter 8, the focus is on production configuration, more specifically the process of configuring routings from a process platform. The fundamental issues in production configuration, including variety handling, routing changes accommodation, levels of abstraction, and constraints satisfaction, are analyzed. To model production configuration while meeting the requirements posed by the fundamental issues, a set of modeling formalism based on nested colored OPNs with changeable structures is developed. In the set of modeling formalism, *PNet*, *ANet*, *MNet*, and *RNet* are defined to represent the conceptual process of the end product, the assembly processes of an assembly family, the manufacturing processes of a part family, and the internal behaviors of a resource object such as a machine, a material handler, and a buffer. The concept of net nesting is introduced to model the complete configuration of routings for given products. In the defined nested net system, the lower level more elaborated nets, for example, *ANets* and *MNets*, are nested in the places of higher level more abstract nets, e.g., *ANets* and *PNet*, as colored tokens. When the lower level nets are nested in the higher level nets, they denote the particular item variants produced

by the defined processes. The communication between nets at two adjacent levels is accomplished by introducing port and socket places, which are defined for resource objects only. Through sending and receiving messages between two resource objects represented by the port and socket places, the nets at the higher level can be activated.

The concept of process platform-based production configuration is tested in an electronics company that produces a high variety of tailor-made vibration motors for hand phones. The results of case studies, along with sensitivity analysis, performance evaluation and system analysis, are discussed in detail, which demonstrate the feasibility and potential of process platform-based production configuration.

9.2. Conclusions

Process platform-based production configuration involves a number of fundamental issues. Accordingly, the research objectives in this dissertation are identified as (1) formulation of process platform, (2) structural representation of process platform, (3) identification of generic routings for process families, (4) identification of mapping relationships between product and process variety, and (5) modeling of production configuration.

The mathematical models in Chapter 4 are established to formulate a process platform with consideration of both static and dynamic behavior. The operation-centered static models and the variety handler and state-based behavioral models of configuring routings provide explicitly and rigorously the basic constructs and a holistic view of a process platform. The process platform's UML models represent a process platform in terms of the constituent elements and their relationships. The structural representation of a process platform is essential to develop solutions for other issues involved in process platform-based production configuration, because the process platform constituent elements and relationships among them are the fundamentals to be understood.

The systematic data mining methodology in Chapter 6 is proposed to identify generic routings for process families for easing process platform construction. The formed generic routings are common to all the individualized routings for producing the corresponding products. The performance evaluation of the proposed methodology is conducted with respect to sensitivity analysis of different threshold values for clustering routings and benchmarking against a manual approach for the formation of generic routings. Sensitivity analysis addresses the tradeoff between the process families to be clustered and the degree of economy of scale in production to be realized. It thus assists companies in making decisions about the number of process platforms to be constructed and the process platforms themselves. The comparison between the data mining methodology and the manual approach shows the advantages of the performance measure of the former over the latter.

In Chapter 7, an association rule mining approach is discussed for identifying mapping relationships between product and process variety in a process platform, which assist variety coordination. Variety coordination facilitates identification of production variations for any changes in design using the mapping relationships. In this regard, the mapping relationships between product and process variety play the role of configuration rules that guide the selection and arrangement of compatible process elements in production configuration. A set of modeling formalism based on nested colored object-oriented PNs with changeable structures is proposed in Chapter 8 for modeling production configuration in terms of the configuration process. Combining the advantages of colored PNs, object PNs, structural change handling mechanisms in PNs, and nesting concept, the developed formalism has the ability to model production configuration accurately. Accordingly, the fundamental issues in production configuration can be captured explicitly, which eventually supports production configuration automation.

9.3. Contributions

The research in this dissertation provides contributions for both manufacturing industries and the academia. Process platform-based production configuration assists companies in implementing mass customization successfully. From the research point of view, it is expected to open up a new research area with respect to process platforms and production configuration. The more specific industrial and academic significance are listed as follows.

(1) *Knowledge Maintenance.* Since a process platform organizes and documents systematically both product and process data, information and knowledge in a single entity, a company's existing knowledge can thus be maintained and updated for future product development in an easy way.

(2) *Effective Variety Management in Mass Customization.* Process platforms are conducive to effective variety management. The generic product and process structures containing all the product and relevant process data assist in managing product and process variety individually, whilst the mapping relationships in between contribute to coordinating product variety in the design domain and process variety in the process domain.

(3) *Realization of the Economy of Scale in Production.* From a process platform, routings are configured by reusing existing knowledge and process elements, including operations, operations precedence and manufacturing resources. The obtained routings will incur the fewest changes in production as possible. Thus, the economy of scale in production is expected to be achieved through production activity repetition.

(4) *Reduction of Product Development Time.* Process platform-based production configuration is able to reduce the time lag between design and production, i.e., time for planning routings, due to the presence of the generic routing structures and the many configuration rules. Consequently, product development time can be reduced.

(5) *Proposal of Process Platform-based Production Configuration.* The abundant research in design and the limited research in routing planning form the potential area of process platforms and production configuration to be explored. It is expected that based on this research, more research and studies can be performed to identify more research problems and provide the corresponding solutions.

(6) *Formulation of Mathematical Models of a Process Platform.* Process platform formulation is accomplished by establishing mathematical models, which describe rigorously the concepts and functionalities of a process platform. Therefore, these models provide the basic constructs and a holistic view of a process platform.

(7) *Construction of Process Platform Structural Models based on UML.* Representing process platform structures using UML clarifies the involved elements and relationships, which are basic to a process platform. The analysis of other issues involved in process platform-based production configuration can thus be accommodated by the structural representation of a process platform.

(8) *Development of a Data Mining Methodology for Generic Routing Identification.* Identification of generic routings is central to process platform construction, since generic routing structures underpin the associated process platforms. Therefore, the data mining methodology sheds light on an effective and efficient approach to constructing process platforms.

(9) *Introduction of an Association Rule Mining Approach for Variety Coordination.* The linchpin of variety coordination in process platforms lies in the mapping relationships between product and process variety. The mapping relationships are exhibited by the correlations between the massive product and process data. The proposed association rule mining approach helps uncover the hidden yet useful correlation patterns behind these

massive data. Thus, variety coordination in process platforms can be facilitated by the proposed association rule mining approach.

(10) Development of a set of Modeling Formalism based on Nested Colored Object-Oriented Petri Nets with Changeable Structures for Production Configuration Modeling. In system modeling, one important issue to be considered is the properness of the modeling tools. Only when the system models are built using the most suitable modeling tools can the models capture and reflect correctly the systems to be analyzed from certain aspects. A set of modeling formalism based on nested colored object-oriented PNs with changeable structure is developed in this research to model production configuration. Taking into account the modeling requirements raised by the fundamental issues in production configuration, the set of modeling formalism has the ability to model production configuration explicitly.

9.4. Limitations

While the research work in this dissertation has significance from both the industrial and research perspectives, some improvements can be made taking into account their abilities of problem solving.

(1) Products applicable to Process Platform-based Production Configuration. Configuring routings from a process platform fits well into the production environment, wherein a variety of mechatronic products are involved. Mechatronic products have clear BOM structures and routings. Assembly operations and sequences play an important role in producing such products, while fabrication operations and processes do not hold the high priority. Therefore, process platform-based production configuration may not guarantee that companies producing mechanical products obtain a competitive advantage in that production of mechanical products depends heavily on manufacturing operations rather than assembly operations.

(2) *Operations Considered in the Data Mining Methodology.* In the proposed data mining methodology, two material items are considered in each assembly operation. While in current production practice, often two components are assembled together, there are cases that more than two components are joined in one operation. Thus, the proposed data mining methodology cannot provide good results when more than two materials are involved in most assembly operations in routings.

9.5. Avenues for Future Research

Based on the understanding of process platform-based production configuration and the work that has been completed, avenues for future research are highlighted next.

(1) *Production Configuration Formulation.* Production configuration formulation aims to define rigorously production configuration so as to ease the development of process platforms and the associated configuration mechanism. As discussed in Chapter 3, it is possible to formulate production configuration as a constraint satisfaction problem.

(2) *Production Configuration Space Modeling.* In configuration automation, the computational efforts for searching possible solutions in the configuration space can be reduced through proper configuration space modeling. Therefore, future efforts should also be placed on the track of production configuration space modeling.

(3) *Production Configuration Evaluation.* In Chapter 6, evaluation has been conducted towards identified generic routings. While the evaluation validates the proposed methodology and specifies the optimal generic routings, it does not focus on evaluation of the configured routings. Thus, more powerful approaches need to be developed to perform evaluation on both the process platforms and the configured routings. Such approaches should possess the ability to judge how many and which generic routings are necessary for a process family as well as which routing configured from the evaluated process platforms is optimal considering the cohort of the family.

(4) *Production Configuration Implementation.* Implementation of process platform-based production configuration is also an important issue to be addressed in the future. The attempt is to provide solutions directing companies to build application systems. In building such application systems, legacy systems, for example, CAPP and CAAP systems, and MRPII systems, may be integrated with process platforms for exchanging the latest data and information. Such data and information can ensure the cost effectiveness and efficiency of the provided solutions.

(5) *Data Mining Methodology Extension.* The developed data mining methodology to identifying generic routings for process families in this research is applicable to operations in which only two material items are involved. Therefore, in the possible future research, the methodology may be improved to handle such assembly operations that are performed to join more than two material items simultaneously. The extension of the data mining methodology is to accommodate more operation types.

REFERENCES

Abdi, M.R., and Labib, A.W., (2003) A design strategy for reconfigurable manufacturing systems (RMSs) using analytical hierarchical process (AHP): A case study. *International Journal of Production Research*, **41**(10), pp. 2273–2299.

Abdi, M.R., and Labib, A.W., (2004) Grouping and selecting products: The design key of reconfigurable manufacturing systems (RMSs). *International Journal of Production Research*, **42**(3), pp. 521–546.

Adamou, M., Zerhouni, S.N., and Bourjault, A., (1998) Hierarchical modeling and control of flexible assembly systems using object-oriented Petri nets. *International Journal of Computer Integrated Manufacturing*, **11**(1), pp. 18–33.

Agard, B., and Kusiak, A., (2004) Data-mining-based methodology for the design of product families. *International Journal of Product Research*, **42**(15), pp. 2955–2969.

Aggarwal, C.C., and Yu, P.S., (1998) Online generation of association rules. *Proceedings of the 14th International Conference on Data Engineering*. pp. 402–411, Orlando, Florida.

Agrawal, R., Imielinski, T., and Swami, A., (1993) Mining association rules between sets of items in massive database. *Proceedings of the ACM/SIGMOD International Conference on Management of Data*, pp. 207–216.

Agrawal, R., and Srikant, R., (1994) Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Databases*, Santiago, Chile, pp. 487–499.

Anand, S.S., and Buchner, A.G., (1998) *Decision Support Using Data Mining*. London: Financial Times Pitman.

- Ardissono, L., and Goy, A., (2000) Tailoring the interaction with users in Web stores. *User Modeling and User-Adapted Interaction*, **10**(4), pp. 251–303.
- Atkinson-Abutridy, J., Mellish, C., and Aitken, S., (2003) A semantically guided and domain-independent evolutionary model for knowledge discovery from texts. *IEEE Transactions on Evolutionary Computation*, **7**(6), pp. 546–560.
- Bain, S., Thornton, J., and Sattar, A., (2004) Evolving algorithms for constraint satisfaction. *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, **1**, pp. 265–272.
- Baker, K.R., Magazine, M.J., and Nuttle, H.L.W., (1986) The effect of commonality on safety stock in a simple inventory model. *Management Science*, **32**(8), pp. 982–988.
- Bauer, B., and Odell, J., (2005) UML 2.0 and agents: How to build agent-based systems with the new UML standard. *Engineering Applications of Artificial Intelligence: Agent-Oriented Software Development*, **18**(2), pp. 141–157.
- Berry, M.J.A., and Linoff, G., (1997) *Data Mining Techniques: For Marketing, Sales, and Customer Support*. New York: Wiley.
- Blackburn, J.H., (1985) Improve MRP and JIT compatibility by combining routings and bills of material. *Proceedings of the American Production and Inventory Control Society*, pp. 444–447.
- Booch, G., (1994) *Object-Oriented Analysis and Design*. Benjamin/Cummings publishing company, Inc.
- Brown, D., (1998) Defining configuring. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **12**(4), pp. 301–305.
- Buchner, A.G., Anand, S.S., and Hugues, J.G., (1997) Data mining in manufacturing environments: Goals, techniques and applications. *Studies in Informatics and Control*, **6**(4), pp. 319–328.

Callahan, S., (2006), Extended generic product structure: An information model for representing product families. *ASME Journal of Computing and Information Science in Engineering*, **6**(3), pp. 263–275

Chakrabarty, S., and Wolter, J., (1997) A structure-oriented approach to assembly sequence planning. *IEEE Transactions on Robotics and Automation*, **13**(1), pp. 14–29.

Chang, T.C., and Wysk, R.A., (1985) *An Introduction to Automated Process Planning Systems*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Chen, M.C., (2003) Configuration of cellular manufacturing systems using association rule induction. *International Journal of Production Research*, **41**(2), pp. 381–395.

Chen, M., Han, J., and Yu, P., (1996) Data mining: An overview from database perspective. *IEEE Transactions on Knowledge and Data Engineering*, **8**(6), pp. 866–883.

Chen, M.S., Park, J.S., and Yu, P.S., (1998) Efficient data mining for path traversal patterns, *IEEE Transactions on Knowledge and Data Engineering*, **10**(2), pp. 209–221.

Chen, W., Rosen, D., Allen J., and Mistree, F., (1994) Modularity and the independence of functional requirements in designing complex systems. *Concurrent Product Design*, **74**, pp. 31–38.

Cheung, D.W., Lee, S.D., and Kao, B., (1997) A general incremental technique for maintaining discovered association rule. *Proceedings of the 5th International Conference on Database Systems for Advanced Applications*, Melbourne, Australia, pp. 185–194.

Chick, S.E., Olsen, T.L., Sethuraman, K., Steckle, K.E., and White III, C.C., (2000) A descriptive multi-attribute model for reconfigurable machining system selection examining buyer-supplier relationships. *International Journal of Agile Management Systems*, **2**(1), pp. 33–48.

Collier, D.A., (1981) The measurement and operating benefits of component part commonality. *Decision Sciences*, **12**(1), pp. 85–96.

- Collier, D.A., (1982) Aggregate safety stock levels and component part commonality. *Management Science*, **28**(22), pp. 1296–1303.
- Corso, M., Muffatto, M., and Verganti, R., (1996) Multi-product innovation: Emerging policies in automotive, motorcycle and earth-moving machinery industries. *Proceedings of the 3rd International Product Development Management Conference on New Approaches to Development and Engineering*, Fontainebleau, France.
- Cooley, R., Tan, P.N., and Srivastave, J., (1999) WebSIFT: The web site information filter system. *Proceedings of the Workshop on Web Usage Analysis and User Profiling (WebKDD99)*, San Diego, CA.
- Das, A., Ng, W.K., and Woon, Y.K., (2001) Rapid association rule mining. *Proceeding of the 10th International Conference on Information and Knowledge Management*, November 5-10, Atlanta, GA, USA, pp. 474–481.
- Dougherty, J., Kohavi, R., and Sahami, M., (1995) Supervised and unsupervised discretization of continuous features. *Proceedings of the 12th International Conference on Machine Learning*, Morgan Kaufmann, pp. 194–202.
- Du, X.H., Jiao, J.X., and Tseng, M.M., (2001) Architecture of product family: Fundamentals and methodology. *Concurrent Engineering: Research and Application*, **9**(4), pp. 309–325.
- Du, X.H., Jiao, J.X., and Tseng, M.M., (2002a) Product family modeling and design support: An approach based on graph rewriting systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **16**(2), pp. 103–120.
- Du, X.H., Jiao, J.X., and Tseng, M.M., (2002b) Graph Grammar Based Product Family Modeling. *Concurrent Engineering: Research and Applications*, **10**(2), pp. 113–127.
- Emerson, C., and Ham, I., (1982) An automated coding and process planning system using a DEC PDP-10. *Computing and Industrial Engineering*, **6**(2), pp. 159–168.

Erickson, R.E., (1988) The state of the art in computer-aided process planning. Donated Report, R88-1937281-1, United Technologies Research Center.

Ericsson, J., Karlsson, P-O., Mercer, G., and Robertson, D., (1996) Sharing parts across car models: Lessons from the manufacturers. *Europe's Automotive Components Business*, 1st Quarter, pp. 150–171.

Farrell, R., and Simpson, T. W., (2003) Product platform design to improve commonality in custom products. *Journal of Intelligent Manufacturing*, **14**(6), pp. 541–556.

Fayyad, U.M., and Irani, K.B., (1993) Multi-interval discretization of continuous valued attributes for classification learning. *Proceedings of the XIII International Joint Conference on Artificial Intelligence (IJCAI-93)*, Chambéry, France, 1993, pp. 1022–1027.

Feldman, R., and Dagan, I., (1995) Knowledge discovery in textual databases (KDT). *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, Montreal, Canada, August 20-21, pp. 112–117.

Fellini, R., Papalambros, P., and Weber, T., (2000) Application of product platform design process to automotive powertrains. *Proceedings of the 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, AIAA-2000-4849.

Feng, S.C., and Song, E.Y., (2002) Preliminary design and manufacturing planning integration using intelligent agents. *Proceedings of the 7th International Conference in Computer Supported Collaborative Work in Design*, Brazil, pp. 270–275.

Ferraro, P., and Godin, C., (2003) Optimal mappings with minimum number of connected components in tree-to-tree comparison problems. *Journal of Algorithms*, **48**(2), pp. 385–406.

Fleischanderl, G., Friedrich, G.E., Haselbock, A., Schreiner, H., and Stumptner, M., (1998) Configuring large systems using generative constraint satisfaction. *IEEE Intelligent Systems*, **13**(4), pp. 59–68.

Forza, C., and Salvador, F., (2002) Product configuration and inter-firm co-ordination: An innovative solution from a small manufacturing enterprise. *Computers in Industry*, **49**(1), pp. 37–46.

Franke, D.W., (1998) Configuration research and commercial solutions. *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, **12**(4), pp. 295–300.

Friedrich, G., and Stumptner, M., (1999) Consistency-based configuration. *Proceedings of AAAI-99 Workshop on Configuration*, June, pp. 35–40.

Fu, X., Budzik, J., and Hammond, K.J., (2000) Mining navigation history for recommendation. *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, New Orleans, LA, pp. 106–112.

Fujita, K., Sakaguchi, H., and Akagi, S., (1999) Product variety deployment and its optimization under modular architecture and module communalization. *Proceedings of ASME Design Engineering Technology Conference*, Paper No. DETC99/DFM-8923.

Gao, J.X., and Bowland, N.W., (2002) A product data management integrated product configuration and assembly process planning environment. *Proceedings of the Institution of Mechanical Engineers, part B, Journal of Engineering Manufacture*, **216**, pp. 407–418.

Gertosio, C., and Dussauchoy, A., (2004) Knowledge discovery from industrial databases. *Journal of Intelligent Manufacturing*, **15**(1), pp. 29–37.

Gonzalez-Zugasti, J.P., Otto, K.N., and Baker, J. D., (2000) Modular platform-based product family design. *Proceedings of ASME Design Engineering Technology Conference*, Paper No. DETC-2000/DAC-14238.

Gottipolu, R.B., and Ghosh, K., (1997) Representation and selection of assembly sequences in computer-aided assembly process planning. *International Journal of Production Research*, **35**(12), pp. 3447–3465.

Groppetti, R., Santucci, A., and Senin, N., (1994) On the application of colored Petri nets to computer-aided assembly planning. *IEEE Symposium on Emerging Technologies and Factory Automation*, pp. 381–387.

Han, J., and Kamber, M., (2001) *Data Mining: Concepts and Techniques*. Morgan-Kaufmann, San Mateo, CA.

Han, J., Pei, J., and Yin, Y., (2000) Mining frequent patterns without candidate generation. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 16-18 May, 2000, Dallas, Texas, USA, pp. 1–12.

Harrison, W., Barton, C., and Raghavachari, M., (2000) Mapping UML designs to Java. *SIGPLAN Notices*, **35**(10), pp. 178–187.

Hastings, N.A.J., and Yeh, C.H., (1992) Bill of manufacture. *Production and Inventory Management Journal*, **33**(4), pp. 27–31.

He, D.W., Strege, B., Tolle, H., and Kusiak, A., (2000) Decomposition in automatic generation of Petri nets for manufacturing system control and scheduling. *International Journal of Production Research*, **38**(6), pp. 1437–1457.

Hearst, M.A., (1997) Text data mining: Issues, techniques, and the relationship to information access. *Presentation Notes for UM/MS workshop on data Mining*, July 1997.

Hegge, H.M.H., (1992) A generic bill-of-material processor using indirect identification of products. *Production Planning & Control*, **3**(3), pp. 336–342.

Hegge, H.M.H., and Wortmann, J.C., (1991) Generic bill-of-material: A new product model. *International Journal of Production Economics*, **23**(1-3), pp. 117–128.

Hidber, C., (1999) Online association rule mining. *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, 1999, PA, Pennsylvania, USA, pp. 145–154.

Hiraishi, K., (2000) A Petri-net-based model for the mathematic analysis of multi-agent systems. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC'2000)*, Oct. 2000, Nashville, TN, 4, pp. 3009–3014.

Hodson, W.K., (1992) *Maynard's Industrial Engineering Handbook*. McGraw-Hill, New York.

Hoffmann, C.M., and O'Donnell, M.J., (1982) Pattern matching in trees. *Journal of the ACM*, **29**(1), pp. 68–95.

Homem de Mello, L.S., and Sanderson, A.C., (1991) A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation*, **7**(2), pp. 228–240

Homem de Mello, L.S., and Sanderson, A.C., (1990) AND/OR Graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, **6**(2), pp. 188–199.

Huang, G.Q., Lau, J.S.K., and Mak, K.L., (2003) The impacts of sharing production information on supply chain dynamics: A review of the literature. *International Journal of Production Research*, **41**(7), pp. 1483–1517.

Jandourek, E., (1996) A model for platform development. *Hewlett-Packard Journal*, Article 6.

Jensen, K., (1992) *Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Vol. 1, Springer-Verlag Berlin Heidelberg, Germany.

Jiang, Z., Zuo, M.J., Tu, P.Y., and Fung, R.Y.K., (1999) Object-oriented Petri nets with changeable structure (OPNs-CS) for production system modeling. *International Journal of Advanced Manufacturing Technology*, **15**(6), pp. 445–459.

Jiao, J., and Tseng, M.M., (1999) A pragmatic approach to product costing based on standard time estimation. *International Journal of Operations and Production Management*, **19**(7), pp. 738–755.

Jiao, J., and Tseng, M.M., (2000) Understanding product family for mass customization by developing commonality indices. *Journal of Engineering Design*, **11**(3), pp. 225–243.

Jiao, J., Tseng, M.M., Ma, Q., and Zou, Y., (2000) Generic bill of materials and operations for high-variety production management. *Concurrent Engineering: Research and Application*, **8**(4), pp. 297–322.

Jiao, J., and Tseng, M.M., (2004) Customizability analysis in design for mass customization. *Computer-Aided Design*, **36**(8), pp. 745–757.

Jiao, J., Zhang, L., and Pokharel, S., (2005a) Coordinating product and process variety for mass customized order fulfillment. *Production Planning and Control*, **16**(6), pp. 608–620.

Jiao, J., Zhang, L., and Pokharel, S., (2007) Process platform planning for variety coordination from design to production in mass customization manufacturing. *IEEE Transactions on Engineering Management*, **54**(1), In print.

Jiao, J., Zhang, L., and Pokharel, S., (2005b) Coordinating product and process variety: An approach based on association rule mining. *Proceedings of the 2005 World Congress on Mass Customization and Personalization*, Hong Kong, September 18–21.

Jiao, J., Zhang, L., and Pokharel, S., (2007) Identifying generic routings for product families in mass customization production: A data mining approach. *Decision Support Systems*, 2007, In Print.

Jiao, J., and Zhang, Y.Y., (2005a) Product portfolio planning with customer-engineering interaction. *IIE Transactions*, **37**(9), pp. 801–814.

Jiao, J., and Zhang, Y.Y., (2005b) Product portfolio identification based on association rule mining. *Computer-Aided Design*, **37**(2), pp. 149–172.

- Kim, K., and Chhajed, D., (2000) Commonality in product design: Cost saving, valuation change and cannibalization. *European Journal of Operational Research*, **125**(3), pp. 602–621.
- Kimour, M.T., and Meslati, D., (2004) An approach to building object models with UML in embedded systems. *Journal of Computing and Information Technology - CIT*, **12**(3), pp. 223–235.
- Klein, R., Buchheit, M., and Nutt, W., (1994) Configuration as model construction: The constructive problem solving approach. *Proceedings of Artificial Intelligence in Design*, Lausanne, Switzerland, pp. 201–218.
- Kneppelt, L.R., (1984) Product structuring considerations for master production scheduling. *Production & Inventory Management*, **25**(1), pp. 83–99.
- Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., and Smith, J., (2002) UML for ontology development. *Knowledge Engineering Review*, **17**(1), pp. 61–64.
- Kolisch, R., (2000) Integration of assembly and fabrication for make-to-order production. *International Journal of Production Economics*, **68**(3), pp. 287–306.
- Koren, R., and Ulsoy, A.G., (1997) Reconfigurable manufacturing systems. Engineering Research Center for Reconfigurable Machining Systems (ERC/RMS) Report 1, The University of Michigan, Ann Arbor, MI.
- Koren, R., Heisel, U., Moriwaki, T., Pritschow, G., Ulsoy, G., and Van Brussel, H., (1999) Reconfigurable manufacturing systems. *Annals of the CIRP*, **48**(2), pp. 527–540.
- Kota, S., and Sethuraman, K., (1998) Managing variety in product families through design for commonality, *Proceedings of the ASME Design Engineering Technical Conference*, Paper No.: DETC98/DTM-5651, Atlanta, GA, USA.
- Kovacs, Z., Le Goff, J.M., and McClatchey, R., (1998) Support for product data from design to production. *Computer Integrated Manufacturing Systems*, **11**(4), pp. 285–290.

Kovacs, z., McClatchey, R., Le Goff, J.M., and Baker, N., (1999) Patterns for integrating manufacturing product and process models. *Proceedings of The 3rd International Conference on Enterprise Distributed Object Computing (EDOC'99)*, September 27-30, 1999, Mannheim, Germany.

Kusiak, A., (2000) Decomposition in data mining: An industrial case study. *IEEE Transactions on Electronics Packaging Manufacturing*, **23**(4), pp. 345–353.

Kusiak, A., (2001) Rough set theory: A data mining tool for semiconductor manufacturing. *IEEE Transactions on Electronics Packaging Manufacturing*, **24**(1), pp. 44–50.

Lea, B.R., and Fredendall, L.D., (2002) The impact of management accounting, product structure, product mix algorithm, and planning horizon on manufacturing performance. *International Journal of Production Economics*, **79**(3), pp. 279–299

Lee, D.L., and Chen, Y.M., (1996) Integrated product and process management. *Integrated Computer-Aided Engineering*, **3**(1), pp. 1–4.

Lee, H.L., and Billington, C., (1994) Designing products and processes for postponement. *Management of Design: Engineering and Management Perspective* (Dasu, S., and Eastman, C., Eds.) pp. 105–122. Boston, MA: Kluwer Academic Publishers.

Lee, H.L., and Tang, C.S., (1997) Modeling the costs and benefits of delayed product differentiation. *Management Science*, **43**(1), pp. 40–53.

Lee, S., (1992) Backward assembly planning with assembly cost analysis. *Proceedings of the IEEE International Conference on Robotics and Automation*, May, pp. 2382–2391.

Lee, S., and Shin, Y.G., (1990) A cooperative planning system for flexible assembly. *Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing*, pp. 306–313.

Lehnerd, A.P., (1987) Revitalizing the manufacture and design of mature global products. *Technology and Global Industry: Companies and Nations in the World Economy* (Guile, B.R., and Brooks, H., Eds.) pp. 49– 64. Washington, DC: National Academy Press.

Leu, S.S., Chen, C.N., and Chang, S.L., (2001) Data mining for tunnel support stability: Neural network approach. *Automation in Construction*, **10**(4), pp. 429–441.

Li, H., and Azarm, S., (2002) An approach for product line design selection under uncertainty and competition. *ASME Journal of Mechanical Design*, **124**(3), pp. 385–392.

Li, J.Y., Sehn, H., and Topor, R., (1999) An adaptive method of numerical attribute merging for quantitative association rule mining. *Proceedings of the 5th International Computer Science Conference on Internet Applications*, Hong Kong, China, 13-15 Dec. 1999, pp. 41–52.

Li, X.O., and Lara-Rosano, F., (1999) Modeling an electronic component manufacturing system using object oriented colored Petri nets. *Proceedings of the 6th IEEE International Conference on Electronics, Circuits, and Systems*, Sep. 1999, Pafos, Cyprus.

Lin, C.T., and Lee, C.S.G., (1996) *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice Hall, Npper Saddle River, NJ, USA.

Lin, W.Y., Alvarez, S., and Ruiz, C., (2002) Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, **6**(1), pp. 83–105.

Lopez-Mellado, E., and Almeyda-Canepa, H., (2003) A three-level net formalism for the modeling of multiple mobile robot systems. *International Journal of Computer Integrated Manufacturing*, **18**(2-3), pp. 122–136.

Magro, D., and Torasso, P., (2003) Decomposition strategies for configuration problems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **17**(1), pp. 51–73.

Martin, M.V., and Ishii, K., (1997) Design for variety: Development of complexity indices and design charts. *Proceedings of the 1996 ASME Design Engineering Technical Conference*, DETC97/DFM4359, 14-17 September, Sacramento, CA, USA.

Martinez, M.T., Favrel, J., and Ghodous, P., (2000) Product family manufacturing plan generation and classification. *Concurrent Engineering: Research and Applications*, **8**(1), pp.12–22.

Mather, H., (1987) *Bills of Materials*. Dow Jones-Irwin, Homewood.

McAdams, D.A., Stone, R.B., and Wood, K.L., (1999) Understanding product similarity using customer needs, *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, DETC99/DTM-5660, Las Vegas, NV, USA.

McDermott, J., (1982) A rule-based configurator of computer systems. *Artificial Intelligence*, **19**(1), pp. 39–88.

McDermott, C. M., and Stock, G. N., (1994) The use of common parts and designs in high-tech industries: A strategic approach. *Production and Inventory Management Journal*, **35**(3), pp. 65–68.

McGrath, M., (1995) *Product Strategy for High-Technology Companies*. New York, NY: Irwin Professional Publishing.

McGuinness, D.L., and Wright, J.R., (1998) An industrial strength description logic-based configurator platform. *IEEE Intelligent Systems*, **13**(4), pp. 69–77.

Mehrabi, M.G., Ulsoy, A.G., and Koren, Y., (2000) Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent Manufacturing*, **11**, pp. 403–419.

Mellor, S.J., (2003) Executable and translatable UML. *Embedded Systems Programming*, **16**(2), pp. 25–30.

Meyer, M.H., and Lehnerd, A.P., (1997) *The power of product platforms: Building value and cost leadership*. New York: Free Press.

Meyer, M.H., Tertzakian, P., and Utterback, J.M., (1997) Metrics for managing research and development in the context of the product family. *Management Science*, **43**(1), pp. 88–111.

Meyer, M., and Utterback, J., (1993) The product family and the dynamics of core capability. *Sloan Management Review*, **34**(3), Spring 1993, pp. 29–47.

Mills, S.F., (1997) A resource-focused framework for process engineering. PhD thesis, Southern Methodist University, Dallas, TX.

Ming, X.G., Mak, K.L., and Yan, J.Q., (1999) A hybrid intelligent inference model for computer aided process planning. *Integrated Manufacturing Systems*, **10**(6), pp. 343–353.

Missikoff, M., (2003) Text mining techniques to automatically enrich a domain ontology. *Applied Intelligence*, **18**(3), pp. 323–340.

Mitrofanov, S.P., (1966) *The Scientific Principles of Group Technology*. National Lending Library Translation, Boston, MA.

Mittal, S., and Falkenhainer, B., (1990) Dynamic constraint satisfaction problems. *Proceedings of AAAI90*, pp. 25–32.

Mittal, S., and Frayman, F., (1989) Towards a generic model of configuration tasks. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, pp. 1395–1401.

Moore, K.E., and Gupta, S.M., (1996) Petri net models of flexible and automated manufacturing systems: A survey. *International Journal of Production Research*, **34**(11), pp. 3001–3036.

Moore, W.L., Louviere, J.J., and Verma, R., (1999) Using conjoint analysis to help design product platforms. *Journal of Product Innovation Management*, **16**(1), pp. 27–39.

- Morandin Jr., O., and Kato, E.R.R., (2003) Virtual Petri nets as a modular modeling method for planning and control tasks of FMS. *Proceedings of the 2003 IEEE International Conference on Systems, Man and Cybernetics*, 5-8 Oct., **2**, pp. 1521–1527.
- Nasukawa, T., and Nagano, T., (2001) Text analysis and knowledge mining system. *IBM Systems Journal*, **40**(4), pp. 967–984.
- Naughton, K., Thornton, E., Kerwin, K., and Dawley, H., (1997) Can Honda build a world car? *Business Week*, **100**(7).
- Neff, M., and Boguraev, B.K., (1989) Dictionaries, dictionary grammars and dictionary entry parsing. *Proceedings of the 27th Annual Meeting on Association Computational Linguistics*, June 1989, Vancouver, Canada, pp. 91–101.
- Nunes, N.J., Selic, B., Silva, A.R., and Alvarez, A.T., (2004) UML Modeling Languages and Applications - UML 2004 Satellite Activities, *Lecture Notes in Computer Science*, 3297, 2004.
- Ou-Yang, C., and Pei, H.N., (1999) Developing a STEP-based integration environment to evaluate the impact of an engineering change on MRP. *International Journal of Advanced Manufacturing Technology*, **15**(11), pp. 769–779.
- Parnas, D.L., Clements, P.C., and Weiss, D.M., (1985) The modular structure of complex systems. *IEEE Transactions on Software Engineering*, **11**(3), pp. 259–265.
- Peterson, J.L., (1977) Petri Nets. *Computing Surveys*, **9**(3), pp. 223–252.
- Peterson, J.L., (1981) *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, NJ.
- Petri, C.A., (1962) Kommunikation mit Automaten, *Schriften des IIM*, No. 2, Bonn.
- Pessina, M.W., and Renner, J.R., (1998) Mass customization at Lutron Electronics - A total company process. *Agility & Global Competition*, **2**(2), pp. 50–57.

Pham, D.T., and Dimov, S.S., (1999) A system for automatic extraction of feature-based assembly information. *Proceedings of Institute of Mechanical Engineers, Part B, Journal of Engineering Manufacturing*, **213**(B1), pp. 97–101.

Phillips, P.H., and Mouleeswaran, C.B., (1985) A knowledge-based approach to generative process planning. *Proceedings of the CASA/SME Autofact 85 Conference*, November.

Philpotts, M., (1996) An introduction to the concepts, benefits and terminology of product data management. *Industrial Management and Data Systems*, **96**(4), pp. 11–17.

Pimmler, T.U., and Eppinger, S.D., (1994) Integration analysis of product decompositions. *Proceedings of ASME Design Theory and Methodology conference*, **68**, pp. 343–351.

Pine, B.J., (1993) *Mass Customization – The New Frontier in Business Competition*. Harvard Business School Press, Boston, MA.

Prasad, B., (1998) Designing products for variety and how to manage complexity. *Journal of Product & Brand Management*, **7**(3), pp. 208–222.

Robertson, D., and Ulrich, K., (1998) Planning for product platforms. *Sloan Management Review*, **(39)**4, pp.19–31.

Rohrdanz, F., Mosemann, H., and Wahl, F.M., (1996) HighLAP: A high level system for generating, representing, and evaluating assembly sequences. *IEEE International Joint Symposium on Intelligence and Systems*, Rockville, Maryland, USA, November 1996, pp. 134–141.

Romanowski, C.J., and Nagi, R., (2005) On comparing bills of materials: A similarity/distance measure for unordered trees. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, **35**(2), pp. 249–260.

Romanowske, C.J., and Nagi, R., (2004) A data mining approach to forming generic bills of materials in support of variant design activities. *ASME Journal of Computing and Information Science in Engineering*, **4**(4), pp. 316–328.

Rothwell, R., and Gardiner, P., (1990) Robustness and product design families. *Design Management: A Handbook of Issues and Methods* (Oakley, M., ed.), Basil Blackwell Inc., Cambridge, MA, pp. 279–292.

Roucoules, L., Salomons, O., and Henri, P., (2003) Process planning as an integration of knowledge in the detailed design phase. *International Journal of Computer Integrated Manufacturing*, **16**(1), pp. 25–37.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., (1991) *Object-Oriented Modeling and Design*. Prentice Hall.

Qian, W.H., Yan, Y.H., Zhao, X.F., and Pagello, E., (1996) List representation of assembly sequences. *Proceedings of the IEEE International Conference on Industrial Technology*, pp. 315–319.

Sabbagh, K., (1996) *Twenty-First Century Jet: The Making and Marketing of the Boeing 777*. New York: Scribner.

Sabin, D., and Freuder, E., (1996) Configuration as composite constraint satisfaction. *Proceedings of Artificial Intelligence and Manufacturing Research Planning Workshop*, pp. 153–161.

Sabin, D., and Weigel, R., (1998) Product configuration frameworks-a survey. *IEEE Intelligent Systems*, **13**(4), pp. 42–49.

Samet, H., (1982) Distance transform for images represented by quadrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, pp. 298–303.

Sanderson, S.W., and Uzumeri, M., (1997) *Managing Product Families*. Chicago, IL: Irwin.

- Sastry, R., and Ranganathan, N., (1995) A VLSI architecture for computing the tree-to-tree distance. *Proceedings of the 1st IEEE Symposium on High-Performance Computer Architecture*, pp. 330–339.
- Sawhney, M.S., (1998) Leveraged high-variety strategies: From portfolio thinking to platform thinking. *Journal of the Academy of Marketing Science*, **26**(1), pp.54–61.
- Schierholt, K., (2001) Process configuration: Combining the principles of product configuration and process planning. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **15**(5), pp. 411–424.
- Schreiber, A.Th., Terpstra, P., Magni, P., and van Velzen, M., (1994) Analyzing and implementing VT using COMMON-KADS. *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based System Workshop*, 3.
- Selonen, P., Koskimies, K., and Sakkinen, M., (2003) Transformations between UML diagrams. *Journal of Database Management*, **14**(3), pp. 37–55.
- Shasha, D., Wang, T.L., Zhang, K.Z., and Shih, F.Y., (1994) Exact and approximate algorithms for unordered tree matching. *IEEE Transactions on Systems, Man, and Cybernetics*, **24**(4), pp. 668–678.
- Siddique, Z., and Rosen, D.W., (2001) On combinatorial design spaces for the configuration design of product families. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **15**(1), pp. 91–108.
- Siddique, Z., Rosen, D.W., and Wang, N., (1998) On the applicability of product variety design concepts to automotive platform commonality. *Proceedings of the ASME Design Engineering Technology Conference*, Paper No. DETC980DTM-5661, Atlanta, GA.
- Simpson, T.W., (2004) Product platform design and customization: Status and promise. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **18**(1), pp. 3–20.

Simpson, T.W., Chen, W., Allen, J.K., and Mistree, F., (1999) Use of the robust concept exploration method to facilitate the design of a family of products. *Simultaneous Engineering: Methodologies and Applications* (Roy, U., Usher, J.M., and Parsaei, H.R., Eds.), pp. 247–278. Amsterdam: Gordon and Breach Science Publishers.

Simpson, T.W., Lanutenschlager, U., and Mistree, F., (1997) Toward mass customization in the age of information: The case for open engineering systems. *The Information Revolution: Present and Future*. (Read, W.H., and Porter, A.L., eds.) Ablex Publishing, Greenwich, CT, pp. 49–71.

Slater, P.J.P., (1999) Pconfig: A Web-based configuration tool for configure-to-order products. *Knowledge-Based Systems*, **12**(5-6), pp. 223–230.

Soininen, T., Tiihonen, J., Mannisto, T., and Sulonen, R., (1998) Towards a general ontology of configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **12**(2), pp. 357–372.

Soininen, T., and Gelle, E., (1999) Dynamic constraint satisfaction in configuration. *Proceedings of the AAAI Workshop*, pp. 95–100.

Soininen, T., Niemela, I., Tiihonen, J., and Sulonen, R., (2000) Unified configuration knowledge representation using weight constraint rules. *Proceedings of ECAI 2000 Configuration Workshop*, pp. 79–84.

Stadzisz, P.C., and Henrioud, J.M., (1995) Integrated design of product families and assembly systems. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1290–1295.

Stonebraker, P.W., (1996) Restructuring the bill of material for productivity: A strategic evaluation of product configuration. *International Journal of Production Economics*, **45**(1-3), pp. 251–260.

Stumptner, M., (1997) An overview of knowledge-based configuration. *AI Communications*, **10**(2), pp. 111–123.

Stumptner, M., and Haselbock, A., (1993) A generative constraint formalism for configuration problems. *Proceedings of the 3rd Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*. **728**, pp. 302–313.

Suh, N.P., (2001) *Axiomatic Design - Advances and Applications*. Oxford University Press, New York.

Svensson, C., (2001) A discussion of future challenges to “built to order” SME’s. Mass customization: A threat or a challenge? *Proceedings of the 4th SME International conference on stimulating Manufacturing Excellence in small and Medium Enterprises*, Aalborg, Denmark, May 14–16, pp. 1-9.

Tai, K.C., (1979) The tree-to-tree correction problem. *Journal of the ACM*, **26**(3), pp. 422–433.

Tan, A.H., (1999) Text mining: The state of the art and the challenges. *Proceedings of the 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 99 Workshop on Knowledge Discovery from advanced Database*, Beijing, pp. 65–70.

Tatsiopoulos, I.P., (1996) On the unification of bills of materials and routings. *Computers in Industry*, **31**(3), pp. 293–304.

Thomas, J.P., Nissanke, N., and Baker, K.D., (1996) A hierarchical Petri net framework for the representation and analysis of assembly. *IEEE Transactions of Robotics and Automation*, **12**(2), pp. 268–279.

Thomas, L.D., (1992) Functional implications of component commonality in operational systems. *IEEE Transactions on Systems, Man and Cybernetics*, **22**(3), pp. 548–551.

- Tielemans, P.F.J., 1995, Lead time performance in manufacturing systems. PhD thesis, Rotterdam School of Management, Eburon, Delft.
- Tonshoff, H-K., and Drabow, G., (2003) Virtual planning of highly flexible and reconfigurable manufacturing systems. *CIRP-Journal of Manufacturing Systems*, **32**(1).
- Turner, W.C., Mize, J.H., Case, K.E., and Nazemetz, J.W., (1992) *Introduction to Industrial and Systems Engineering*. Prentice Hall, Englewood Cliffs, NJ.
- Ulrich, K., and Eppinger, S.D., (1995) *Product Design and Development*. McGraw-Hill, Inc.
- Ulrich, K., and Tung, K., (1991) Fundamentals of product modularity. *Proceedings of ASME Winter Annual Meeting Conference*, Atlanta, GA, **39**, pp. 73–80.
- Vakharia, A.J., Parmenter, D.A., and Sanchez, S.M., (1996) The operating impact of parts commonality. *Journal of Operations Management*, **14**(1), pp. 3–18.
- Valiente, G., (2002) *Algorithms on Trees and Graphs*. Springer-Verlag, Berlin.
- van Veen, E.A., (1992) *Modeling Product Structures by Generic Bills-of-Materials*. Elsevier, New York.
- van Veen, E.A., and Wortmann, J.C., (1992) Generative bill of material processing systems. *Production Planning & Control*, **3**(3), pp. 314–326.
- van Vuuren, W., and Halman, J.I.M., (2001) Platform-driven development of product families: Linking theory with practice. *Proceedings of the Conference of the Future of Innovation Studies*, The Netherlands, September 20–23.
- Visa, A., (2001) Technology of text mining. *Proceedings of Machine Learning and Data Mining in Pattern Recognition*, MLDM 2001, pp. 1–11.
- Waldman, H., (1983) Process planning at Sikorsky. *CAD/CAM Technology*, Summer 1983, **13**.

Wang, A., Koc, B., and Nagi, R., (2005a) Complex assembly variant design in agile manufacturing. Part I: System architecture and assembly modeling methodology. *IIE Transactions on Design and Manufacturing*, **37**(1), pp. 1–16.

Wang, A., Koc, B., and Nagi, R., (2005b) Complex assembly variant design in agile manufacturing. Part II: Variant design methodology. *IIE Transactions on Design and Manufacturing*, **37**(1), pp. 17–34.

Wang, E., and Kim, Y.S., (1999) Feature-based assembly mating reasoning. *Journal of Manufacturing Systems*, **18**(3).

Wang, H., and McCauley-Bell, P., (1996) Fuzzy clustering analysis and multifactorial evaluation for students' imaginative power in physics problem solving. *Fuzzy Sets and Systems*, **78**(1), pp. 95–105.

Wang, L.C., (1996a) Object-oriented Petri nets for modeling and analysis of automated manufacturing systems. *Computer Integrated Manufacturing Systems*, **26**(2), pp. 111–125.

Wang, L.C., (1996b) The development of an object-oriented Petri net cell control model. *International Journal of Advanced Manufacturing Technology*, **11**, pp. 59–69.

Wang, L.C., (1996c) An integrated object-oriented Petri net paradigm for manufacturing control systems. *International Journal of Computer Integrated Manufacturing*, **9**(1), pp. 73–87.

Wang, L.C., and Wu, S.Y., (1998) Modeling with colored timed object-oriented Petri nets for automated manufacturing systems. *Computers and Industrial Engineering*, **34**(2), pp. 463–480.

Wang, T.L., Zhang, K.Z., Jeong, K.J., and Shasha, D., (1991) A tool for tree pattern matching. *Proceedings of the 1991 IEEE International Conference on Tools for AI*, San Jose, CA, November, 1991, pp. 436–444.

Weng, S.S., and Lin, Y.J., (2003) A study on searching for similar documents based on multiple concepts and distribution of concepts. *Expert Systems with Applications*, **25**(3), pp. 355–368.

Weng, S.S., and Liu, C.K., (2004) Using text classification and multiple concepts to answer e-mails. *Expert Systems with Applications*, **26**(4), pp. 529–543.

Westphal, C., and Blaxton, T., (1998) *Data Mining Solutions*. New York: Wiley.

Wheelwright, S. C., and Clark, K. B., (1992) *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency and Quality*. New York: The Free Press.

Wheelwright, S.C., and Clark, K.B., (1995) *Leading Product Development*. New York: Free Press.

Whitney, D.E., (1993) Nippondenso Co. Ltd: A case study of strategic product design. *Research in Engineering Design*, **5**(1), pp. 1–20.

Wolter, J.D., and Chandrasekaran, P., (1991) Knowledge representation for design of mechanical assemblies. *Proceedings of the 7th Annual NSF Conference of Design Manufacturing Systems Research*, January 1991, pp. 633–640.

Wortmannm, J.C., Muntslag, D.R., and Timmermans, P.J.M., (1997) *Customer-Driven Manufacturing*. Chapman and Hall, London.

Wu, Z.B., and Wu, C., (1998) An engineering framework system to support integrated product and process development. *High Technology Letters*, **4**(1), pp. 1–8.

Wysk, R.A., (1977) An Automated Process Planning and Selection Program: APPAS, Ph.D. Thesis, Purdue University, West Lafayette, Indiana.

Yang, L., and Gao, Y., (1996) *Fuzzy Mathematics: Theory and Applications*. ISBN. 7-5623-0440-8.

Yang, H.C., and Lee, C.H., (2004) A text mining approach on automatic generation of web directories and hierarchies. *Expert Systems with Applications*, **27**(4), pp. 645–663.

Zha, X.F., (2000) An object-oriented knowledge based Petri net approach to intelligent integration of design and assembly planning. *Artificial Intelligence in Engineering, Design, Analysis and Manufacturing*, **14**(1), pp. 83–112.

Zha, X.F., and Du, H., (2002) A PDES/STEP-based model and system for concurrent integrated design and assembly planning. *Computer-Aided Design*, **34**(14), pp. 1087–1110.

Zhang, J., Chan, F.T.S., Li, P.G., Lau, H.C.W., IP, R.W.L., and Samaranayak, P., (2002) Investigation of the reconfigurable control system for an agile manufacturing cell. *International Journal of Production Research*, **40**(15), pp. 3709–3723.

Zhao, X.B., Wang, J.C., and Luo, Z.B., (2000) A stochastic model of a reconfigurable manufacturing system Part 1: A framework. *International Journal of Production Research*, **38**(10), pp. 2273–2258.

Zhou, J.H., (2003) Discovering association rules in engineering document. *Proceedings of the 2003 International Conference on Natural Language Processing and Knowledge Engineering*, 26-29 Oct. pp. 339–344.

Zimmermann, H.J., (2001) *Fuzzy Set Theory and Its Applications*. 2nd Edition, Kluwer Academic Publishers, USA.

APPENDIX A:

NOMENCLATURE

ABS: Assembly breakdown structures

ACAPS: Automated coding and process planning selection

AGV: Automated guided vehicle

APPAS: Automated process planning and selection

BOM: Bill of materials

BOMfr: Bill of manufacture

CAAP: Computer-aided assembly planning

CAPP: Computer-aided process planning

CMPP: Computer managed process planning

CPN: Colored PN

CSP: Constraint satisfaction problems

ERC/RMS: Engineering research center for reconfigurable manufacturing systems

FR: Functional requirements

GBOM: Generic BOM

GBOMO: Generic bill-of-materials-and-operations

GPcS: Generic process structure

GPdS: Generic product structure

GR: Generic routing

GRS: Generic routing structure

I: Item

IDEF: Integrated definition comprising IDEF0, IDEF1, IDEF1X, IDEF2, IDEF3, IDEF4, and

IDEF5

IF: Intermediate form

ISO: International organization standard

MAPS: Manufacturing assembly process sequencer

MBOM: Modular BOM

MHS: Material handling system

MRP: Material requirements planning

NNSys: Nested net system

OMG: Object management group

OO: Object-oriented

OOCPN: Object-oriented colored PNs

OPN: Object-oriented PN

PBOM: Percentage BOM

PBS: Product breakdown structures

PDM: Product data management

PN: Petri net

PP: Port place

RMS: Reconfigurable manufacturing system

R1/XCON: A production-rule-based system for selection the computer system components

SBOM: Super BOM

SP: Socket place

T: Transaction

TA: Trust area

TDB: Database of transactions

TID: Identifier of a transaction

UML: Unified modeling language

VBOM: Variant BOM

VPN: Virtual PN

WBS: Work breakdown structures

WfMS: Workflow management systems

WIP: Work in process

XTUML: Executable and translatable UML

APPENDIX B:

VIBRATION MOTOR ROUTING DATA

This appendix present details of 30 routings of vibration motor variants. Each routing is presented as a precedence graph along with specific descriptions of operations details.

Figure B.1 Motor routing #1

Table B.1 Operations in motor routing #1

Node	Operation
FmA1	Final motor assembly operation variant 1
MaA1	Mainbody A assembly operation variant 1
RhS5	Rubber holder shaping operation variant 5
WtM2	Weight machining operation variant 2
MbA7	Mainbody B assembly operation variant 7
BaA3	Bracket assembly operation variant 3
FaA4	Frame assembly operation variant 4
BabA4	Subbracket assembly operation variant 4
TIM2	Terminal machining operation variant 2
AaA3	Armature assembly operation variant 3
MtaA9	Magnet assembly operation variant 9
BaM5	Bracket A machining operation variant 5
BbM2	Bracket B machining operation variant 2
CtaA2	Commutator assembly operation variant 2
ClaA4	Coil assembly operation variant 4
FmM1	Frame machining operation variant 1
MhaA7	Magnet housing assembly operation variant 7
MtM7	Magnet machining operation variant 7
MhS1	Magnet housing stamping operation variant 1
CtM1	Commutator machining operation variant 1
StM5	Shaft machining operation variant 5
CIF3	Coil forming operation variant 3
TpS2	Tape shaping operation variant 2

Figure B.2 Motor routing #2

Table B.2 Operations in motor routing #2

Node	Operation
FmA30	Final motor assembly operation variant 30
BaA21	Bracket assembly operation variant 21
BaM18	Bracket A machining operation variant 18
BbM3	Bracket B machining operation variant 3
FaA30	Frame assembly operation variant 30
MtaA23	Magnet assembly operation variant 23
FmM1	Frame machining operation variant 1
MhaA18	Magnet housing assembly operation variant 18
MtM6	Magnet machining operation variant 6
MhS4	Magnet housing stamping operation variant 4
AaA27	Armature assembly operation variant 27
CtaA1	Commutator assembly operation variant 1
ClaA4	Coil assembly operation variant 4
CtM3	Commutator machining operation variant 3
StM4	Shaft machining operation variant 4
CIF3	Coil forming operation variant 3
TpS2	Tape shaping operation variant 2

Table B.3 Operations in motor routing #3

Figure B.3 Motor routing #3

Node	Operation
FmA3	Final motor assembly operation variant 3
MaA7	Mainbody A assembly operation variant 7
RhS8	Rubber holder shaping operation variant 8
WtM4	Weight machining operation variant 4
MbA10	Mainbody B assembly operation variant 10
BaA15	Bracket assembly operation variant 15
FaA18	Frame assembly operation variant 18
BabA2	Subbracket assembly operation variant 2
TIM1	Terminal machining operation variant 1
AaA12	Armature assembly operation variant 12
MtaA13	Magnet assembly operation variant 13
BaM6	Bracket A machining operation variant 6
BbM14	Bracket B machining operation variant 14
CtaA6	Commutator assembly operation variant 6
ClaA9	Coil assembly operation variant 9
FmM4	Frame machining operation variant 4
MhaA10	Magnet housing assembly operation variant 10
MtM3	Magnet machining operation variant 3
MhS7	Magnet housing stamping operation variant 7
CtM5	Commutator machining operation variant 5
StM2	Shaft machining operation variant 2
ClF2	Coil forming operation variant 2
TpS3	Tape shaping operation variant 3

Table B.4 Operations in motor routing #4

Figure B.4 Motor routing #4

Node	Operation
FmA4	Final motor assembly operation variant 4
MbA4	Mainbody B assembly operation variant 4
WtM7	Weight machining operation variant 7
BaA8	Bracket assembly operation variant 8
TIM1	Terminal machining operation variant 1
BabA6	Subbracket assembly operation variant 6
BaM4	Bracket A machining operation variant 4
BbM6	Bracket B machining operation variant 6
FaA1	Frame assembly operation variant 1
RaA1	WArmature assembly operation variant 1
AaA7	Armature assembly operation variant 7
WsM4	Washer machining operation variant 4
MtaA6	Magnet assembly operation variant 6
CtaA8	Commutator assembly operation variant 8
ClaA5	Coil assembly operation variant 5
FmM4	Frame machining operation variant 4
MhaA9	Magnet housing assembly operation variant 9
MtM7	Magnet machining operation variant 7
MhS2	Magnet housing stamping operation variant 2
CtM9	Commutator machining operation variant 9
StM7	Shaft machining operation variant 7
ClF6	Coil forming operation variant 6
TpS2	Tape shaping operation variant 2

Table B.5 Operations in motor routing #5

Figure B.5 Motor routing #5

Node	Operation
FmA5	Final motor assembly operation variant 5
MbA9	Mainbody B assembly operation variant 9
RhS2	Rubber holder shaping operation variant 2
BaA11	Bracket assembly operation variant 11
TIM2	Terminal machining operation variant 2
BabA14	Subbracket assembly operation variant 14
BaM13	Bracket A machining operation variant 13
BbM5	Bracket B machining operation variant 5
FaA12	Frame assembly operation variant 12
RaA7	WArmature assembly operation variant 7
AaA11	Armature assembly operation variant 11
WsM6	Washer machining operation variant 6
MtaA14	Magnet assembly operation variant 14
CtaA15	Commutator assembly operation variant 15
ClaA10	Coil assembly operation variant 10
FmM5	Frame machining operation variant 5
MhaA11	Magnet housing assembly operation variant 11
MtM4	Magnet machining operation variant 4
MhS7	Magnet housing stamping operation variant 7
CtM4	Commutator machining operation variant 4
StM4	Shaft machining operation variant 4
ClF4	Coil forming operation variant 4
TpS8	Tape shaping operation variant 8

Table B.6 Operations in motor routing #6

Figure B.6 Motor routing #6

Node	Operation
FmA6	Final motor assembly operation variant 6
MbA8	Mainbody B assembly operation variant 8
RhS4	Rubber holder shaping operation variant 4
BaA1	Bracket assembly operation variant 1
TIM4	Terminal machining operation variant 4
BabA3	Subbracket assembly operation variant 3
BaM7	Bracket A machining operation variant 7
BbM14	Bracket B machining operation variant 14
FaA5	Frame assembly operation variant 5
RaA6	WArmature assembly operation variant 6
AaA4	Armature assembly operation variant 4
WsM7	Washer machining operation variant 7
MtaA3	Magnet assembly operation variant 3
CtaA1	Commutator assembly operation variant 1
ClaA8	Coil assembly operation variant 8
FmM9	Frame machining operation variant 9
MhaA4	Magnet housing assembly operation variant 4
MtM3	Magnet machining operation variant 3
MhS5	Magnet housing stamping operation variant 5
CtM3	Commutator machining operation variant 3
StM4	Shaft machining operation variant 4
ClF5	Coil forming operation variant 5
TpS1	Tape shaping operation variant 1

Table B.7 Operations in motor routing #7

Figure B.7 Motor routing #7

Node	Operation
FmA7	Final motor assembly operation variant 7
MbA12	Mainbody B assembly operation variant 12
WtM3	Weight machining operation variant 3
BaA18	Bracket assembly operation variant 18
TIM5	Terminal machining operation variant 5
BabA1	Subbracket assembly operation variant 1
BaM12	Bracket A machining operation variant 12
BbM2	Bracket B machining operation variant 2
FaA14	Frame assembly operation variant 14
RaA8	WArmature assembly operation variant 8
AaA17	Armature assembly operation variant 17
WsM5	Washer machining operation variant 5
MtaA12	Magnet assembly operation variant 12
CtaA7	Commutator assembly operation variant 7
ClaA11	Coil assembly operation variant 11
FmM3	Frame machining operation variant 3
MhaA16	Magnet housing assembly operation variant 16
MtM3	Magnet machining operation variant 3
MhS9	Magnet housing stamping operation variant 9
CtM6	Commutator machining operation variant 6
StM1	Shaft machining operation variant 1
ClF3	Coil forming operation variant 3
TpS1	Tape shaping operation variant 1

Table B.8 Operations in motor routing #8

Figure B.8 Motor routing #8

Node	Operation
FmA8	Final motor assembly operation variant 8
MbA11	Mainbody B assembly operation variant 11
RhS3	Rubber holder shaping operation variant 3
BaA14	Bracket assembly operation variant 14
TIM6	Terminal machining operation variant 6
BabA9	Subbracket assembly operation variant 9
BaM8	Bracket A machining operation variant 8
BbM15	Bracket B machining operation variant 15
FaA13	Frame assembly operation variant 13
RaA5	WArmature assembly operation variant 5
AaA13	Armature assembly operation variant 13
WsM3	Washer machining operation variant 3
MtaA17	Magnet assembly operation variant 17
CtaA12	Commutator assembly operation variant 12
ClaA15	Coil assembly operation variant 15
FmM2	Frame machining operation variant 2
MhaA13	Magnet housing assembly operation variant 13
MtM8	Magnet machining operation variant 8
MhS3	Magnet housing stamping operation variant 3
CtM1	Commutator machining operation variant 1
StM7	Shaft machining operation variant 7
ClF4	Coil forming operation variant 4
TpS5	Tape shaping operation variant 5

Figure B.9 Motor routing #9

Table B.9 Operations in motor routing #9

Node	Operation
FmA9	Final motor assembly operation variant 9
MbA3	Mainbody B assembly operation variant 3
WtM5	Weight machining operation variant 5
BaA9	Bracket assembly operation variant 9
BaM10	Bracket A machining operation variant 10
BbM1	Bracket B machining operation variant 1
FaA7	Frame assembly operation variant 7
RaA2	WArmature assembly operation variant 2
AaA5	Armature assembly operation variant 5
WsM1	Washer machining operation variant 1
MtaA4	Magnet assembly operation variant 4
CtaA3	Commutator assembly operation variant 3
ClaA7	Coil assembly operation variant 7
FmM5	Frame machining operation variant 5
MhaA7	Magnet housing assembly operation variant 7
MtM2	Magnet machining operation variant 2
MhS4	Magnet housing stamping operation variant 4
CtM2	Commutator machining operation variant 2
StM3	Shaft machining operation variant 3
ClF10	Coil forming operation variant 10
TpS4	Tape shaping operation variant 4

Table B.10 Operations in motor routing #10

Figure B.10 Motor routing #10

Node	Operation
FmA10	Final motor assembly operation variant 10
RhS1	Rubber holder shaping operation variant 1
MbA2	Mainbody B assembly operation variant 2
BaA5	Bracket assembly operation variant 5
FaA3	Frame assembly operation variant 3
BabA7	Subbracket assembly operation variant 7
TIM3	Terminal machining operation variant 3
AaA1	Armature assembly operation variant 1
MtaA5	Magnet assembly operation variant 5
BaM3	Bracket A machining operation variant 3
BbM19	Bracket B machining operation variant 19
CtaA4	Commutator assembly operation variant 4
ClaA9	Coil assembly operation variant 9
FmM10	Frame machining operation variant 10
MhaA2	Magnet housing assembly operation variant 2
MtM5	Magnet machining operation variant 5
MhS3	Magnet housing stamping operation variant 3
CtM4	Commutator machining operation variant 4
StM10	Shaft machining operation variant 10
ClF2	Coil forming operation variant 2
TpS3	Tape shaping operation variant 3

Figure B.11 Motor routing #11

Table B.11 Operations in motor routing #11

Node	Operation
FmA11	Final motor assembly operation variant 11
MbA1	Mainbody B assembly operation variant 1
RhS3	Rubber holder shaping operation variant 3
BaA7	Bracket assembly operation variant 7
BaM6	Bracket A machining operation variant 6
BbM3	Bracket B machining operation variant 3
FaA2	Frame assembly operation variant 2
RaA4	WArmature assembly operation variant 4
AaA2	Armature assembly operation variant 2
WsM2	Washer machining operation variant 2
MtaA10	Magnet assembly operation variant 10
CtaA6	Commutator assembly operation variant 6
ClaA10	Coil assembly operation variant 10
FmM7	Frame machining operation variant 7
MhaA6	Magnet housing assembly operation variant 6
MtM4	Magnet machining operation variant 4
MhS9	Magnet housing stamping operation variant 9
CtM5	Commutator machining operation variant 5
StM2	Shaft machining operation variant 2
ClF4	Coil forming operation variant 4
TpS8	Tape shaping operation variant 8

Figure B.12 Motor routing #12

Table B.12 Operations in motor routing #12

Node	Operation
FmA12	Final motor assembly operation variant 12
BaA2	Bracket assembly operation variant 2
BabA6	Subbracket assembly operation variant 6
TIM1	Terminal machining operation variant 1
BaM2	Bracket A machining operation variant 2
BbM15	Bracket B machining operation variant 15
FaA10	Frame assembly operation variant 10
RaA9	WArmature assembly operation variant 9
AaA9	Armature assembly operation variant 9
WsM5	Washer machining operation variant 5
MtaA2	Magnet assembly operation variant 2
CtaA5	Commutator assembly operation variant 5
ClaA1	Coil assembly operation variant 1
FmM6	Frame machining operation variant 6
MhaA3	Magnet housing assembly operation variant 3
MtM6	Magnet machining operation variant 6
MhS7	Magnet housing stamping operation variant 7
CtM8	Commutator machining operation variant 8
StM9	Shaft machining operation variant 9
ClF7	Coil forming operation variant 7
TpS9	Tape shaping operation variant 9

Figure B.13 Motor routing #13

Table B.13 Operations in motor routing #13

Node	Operation
FmA13	Final motor assembly operation variant 13
MaA3	Mainbody A assembly operation variant 3
RhS2	Rubber holder shaping operation variant 2
WtM7	Weight machining operation variant 7
MbA5	Mainbody B assembly operation variant 5
BaA6	Bracket assembly operation variant 6
BaM10	Bracket A machining operation variant 10
BbM18	Bracket B machining operation variant 18
FaA8	Frame assembly operation variant 8
AaA8	Armature assembly operation variant 8
MtaA26	Magnet assembly operation variant 26
CtaA10	Commutator assembly operation variant 10
ClaA2	Coil assembly operation variant 2
FmM8	Frame machining operation variant 8
MhaA1	Magnet housing assembly operation variant 1
MtM8	Magnet machining operation variant 8
MhS8	Magnet housing stamping operation variant 8
CtM7	Commutator machining operation variant 7
StM8	Shaft machining operation variant 8
ClF1	Coil forming operation variant 1
TpS6	Tape shaping operation variant 6

Figure B.14 Motor routing #14

Table B.14 Operations in motor routing #14

Node	Operation
FmA14	Final motor assembly operation variant 14
MaA5	Mainbody A assembly operation variant 5
RhS4	Rubber holder shaping operation variant 4
WtM2	Weight machining operation variant 2
MbA15	Mainbody B assembly operation variant 15
BaA23	Bracket assembly operation variant 23
BaM11	Bracket A machining operation variant 11
BbM17	Bracket B machining operation variant 17
FaA15	Frame assembly operation variant 15
AaA20	Armature assembly operation variant 20
MtaA14	Magnet assembly operation variant 14
CtaA8	Commutator assembly operation variant 8
ClaA4	Coil assembly operation variant 4
FmM5	Frame machining operation variant 5
MhaA11	Magnet housing assembly operation variant 11
MtM4	Magnet machining operation variant 4
MhS7	Magnet housing stamping operation variant 7
CtM9	Commutator machining operation variant 9
StM7	Shaft machining operation variant 7
ClF3	Coil forming operation variant 3
TpS2	Tape shaping operation variant 2

Table B.15 Operations in motor routing #15

Figure B.15 Motor routing #15

Node	Operation
FmA15	Final motor assembly operation variant 15
BaA20	Bracket assembly operation variant 20
BabA10	Subbracket assembly operation variant 10
TIM4	Terminal machining operation variant 4
BaM2	Bracket A machining operation variant 2
BbM4	Bracket B machining operation variant 4
FaA20	Frame assembly operation variant 20
RaA10	WArmature assembly operation variant 10
AaA18	Armature assembly operation variant 18
WsM4	Washer machining operation variant 4
MtaA16	Magnet assembly operation variant 16
CtaA13	Commutator assembly operation variant 13
ClaA12	Coil assembly operation variant 12
FmM4	Frame machining operation variant 4
MhaA17	Magnet housing assembly operation variant 17
MtM7	Magnet machining operation variant 7
MhS5	Magnet housing stamping operation variant 5
CtM3	Commutator machining operation variant 3
StM5	Shaft machining operation variant 5
ClF4	Coil forming operation variant 4
TpS2	Tape shaping operation variant 2

Table B.16 Operations in motor routing #16

Figure B.16 Motor routing #16

Node	Operation
FmA16	Final motor assembly operation variant 16
MbA14	Mainbody B assembly operation variant 14
RhS7	Rubber holder shaping operation variant 7
BaA22	Bracket assembly operation variant 22
BaM3	Bracket A machining operation variant 3
BbM11	Bracket B machining operation variant 11
FaA17	Frame assembly operation variant 17
RaA13	WArmature assembly operation variant 13
AaA21	Armature assembly operation variant 21
WsM5	Washer machining operation variant 5
MtaA14	Magnet assembly operation variant 14
CtaA20	Commutator assembly operation variant 20
ClaA16	Coil assembly operation variant 16
FmM3	Frame machining operation variant 3
MhaA14	Magnet housing assembly operation variant 14
MtM5	Magnet machining operation variant 5
MhS5	Magnet housing stamping operation variant 5
CtM6	Commutator machining operation variant 6
StM3	Shaft machining operation variant 3
ClF8	Coil forming operation variant 8
TpS5	Tape shaping operation variant 5

Figure B.17 Motor routing #17

Table B.17 Operations in motor routing #17

Node	Operation
FmA17	Final motor assembly operation variant 17
RhS1	Rubber holder shaping operation variant 1
MbA13	Mainbody B assembly operation variant 13
BaA27	Bracket assembly operation variant 27
FaA23	Frame assembly operation variant 23
BabA12	Subbracket assembly operation variant 12
TIM5	Terminal machining operation variant 5
AaA16	Armature assembly operation variant 16
MtaA19	Magnet assembly operation variant 19
BaM5	Bracket A machining operation variant 5
BbM6	Bracket B machining operation variant 6
CtaA15	Commutator assembly operation variant 15
ClaA1	Coil assembly operation variant 1
FmM8	Frame machining operation variant 8
MhaA20	Magnet housing assembly operation variant 20
MtM3	Magnet machining operation variant 3
MhS1	Magnet housing stamping operation variant 1
CtM4	Commutator machining operation variant 4
StM4	Shaft machining operation variant 4
ClF7	Coil forming operation variant 7
TpS9	Tape shaping operation variant9

Figure B.18 Motor routing #18

Table B.18 Operations in motor routing #18

Node	Operation
FmA18	Final motor assembly operation variant 18
MbA17	Mainbody B assembly operation variant 17
WtM5	Weight machining operation variant 5
BaA29	Bracket assembly operation variant 29
BaM17	Bracket A machining operation variant 17
BbM9	Bracket B machining operation variant 9
FaA29	Frame assembly operation variant 29
RaA15	WArmature assembly operation variant 15
AaA29	Armature assembly operation variant 29
WsM8	Washer machining operation variant 8
MtaA20	Magnet assembly operation variant 20
CtaA14	Commutator assembly operation variant 14
ClaA18	Coil assembly operation variant 18
FmM6	Frame machining operation variant 6
MhaA25	Magnet housing assembly operation variant 25
MtM2	Magnet machining operation variant 2
MhS9	Magnet housing stamping operation variant 9
CtM7	Commutator machining operation variant 7
StM3	Shaft machining operation variant 3
ClF10	Coil forming operation variant 10
TpS7	Tape shaping operation variant 7

Figure B.19 Motor routing #19

Table B.19 Operations in motor routing #19

Node	Operation
FmA19	Final motor assembly operation variant 19
BaA10	Bracket assembly operation variant 10
BaM9	Bracket A machining operation variant 9
BbM7	Bracket B machining operation variant 7
FaA9	Frame assembly operation variant 9
RaA3	WArmature assembly operation variant 3
AaA6	Armature assembly operation variant 6
WsM3	Washer machining operation variant 3
MtaA8	Magnet assembly operation variant 8
CtaA9	Commutator assembly operation variant 9
ClaA3	Coil assembly operation variant 3
FmM4	Frame machining operation variant 4
MhaA10	Magnet housing assembly operation variant 10
MtM9	Magnet machining operation variant 9
MhS10	Magnet housing stamping operation variant 10
CtM4	Commutator machining operation variant 4
StM6	Shaft machining operation variant 6
ClF9	Coil forming operation variant 9
TpS5	Tape shaping operation variant 5

Figure B.20 Motor routing #20

Table B.20 Operations in motor routing #20

Node	Operation
FmA20	Final motor assembly operation variant 20
MbA16	Mainbody B assembly operation variant 16
WtM3	Weight machining operation variant 3
BaA19	Bracket assembly operation variant 19
BaM14	Bracket A machining operation variant 14
BbM5	Bracket B machining operation variant 5
FaA18	Frame assembly operation variant 18
AaA22	Armature assembly operation variant 22
MtaA6	Magnet assembly operation variant 6
CtaA16	Commutator assembly operation variant 16
ClaA11	Coil assembly operation variant 11
FmM3	Frame machining operation variant 3
MhaA9	Magnet housing assembly operation variant 9
MtM7	Magnet machining operation variant 7
MhS2	Magnet housing stamping operation variant 2
CtM8	Commutator machining operation variant 8
StM4	Shaft machining operation variant 4
ClF3	Coil forming operation variant 3
TpS1	Tape shaping operation variant 1

Figure B.21 Motor routing #21

Table B.21 Operations in motor routing #21

Node	Operation
FmA21	Final motor assembly operation variant 21
BaA4	Bracket assembly operation variant 4
BaM4	Bracket A machining operation variant 4
BbM12	Bracket B machining operation variant 12
FaA21	Frame assembly operation variant 21
RaA12	WArmature assembly operation variant 12
AaA24	Armature assembly operation variant 24
WsM2	Washer machining operation variant 2
MtaA5	Magnet assembly operation variant 5
CtaA8	Commutator assembly operation variant 8
ClaA20	Coil assembly operation variant 20
FmM10	Frame machining operation variant 10
MhaA2	Magnet housing assembly operation variant 2
MtM5	Magnet machining operation variant 5
MhS3	Magnet housing stamping operation variant 3
CtM9	Commutator machining operation variant 9
StM7	Shaft machining operation variant 7
ClF6	Coil forming operation variant 6
TpS8	Tape shaping operation variant 8

Figure B.22 Motor routing #22

Table B.22 Operations in motor routing #22

Node	Operation
FmA22	Final motor assembly operation variant 22
MbA19	Mainbody B assembly operation variant 19
WtM4	Weight machining operation variant 4
BaA28	Bracket assembly operation variant 28
BaM15	Bracket A machining operation variant 15
BbM8	Bracket B machining operation variant 8
FaA28	Frame assembly operation variant 28
AaA30	Armature assembly operation variant 30
MtaA2	Magnet assembly operation variant 2
CtaA17	Commutator assembly operation variant 17
ClaA19	Coil assembly operation variant 19
FmM6	Frame machining operation variant 6
MhaA3	Magnet housing assembly operation variant 3
MtM6	Magnet machining operation variant 6
MhS7	Magnet housing stamping operation variant 7
CtM3	Commutator machining operation variant 3
StM8	Shaft machining operation variant 8
ClF5	Coil forming operation variant 5
TpS5	Tape shaping operation variant 5

Table B.23 Operations in motor routing #23

Figure B.23 Motor routing #23

Node	Operation
FmA23	Final motor assembly operation variant 23
BaA16	Bracket assembly operation variant 16
BabA11	Subbracket assembly operation variant 11
TIM3	Terminal machining operation variant 3
BaM19	Bracket A machining operation variant 19
BbM4	Bracket B machining operation variant 4
FaA22	Frame assembly operation variant 22
AaA14	Armature assembly operation variant 14
MtaA18	Magnet assembly operation variant 18
FmM9	Frame machining operation variant 9
MhaA21	Magnet housing assembly operation variant 21
MtM7	Magnet machining operation variant 7
MhS8	Magnet housing stamping operation variant 8
CtaA11	Commutator assembly operation variant 11
ClaA8	Coil assembly operation variant 8
CtM4	Commutator machining operation variant 4
StM3	Shaft machining operation variant 3
ClF5	Coil forming operation variant 5
TpS1	Tape shaping operation variant 1

Table B.24 Operations in motor routing #24

Figure B.24 Motor routing #24

Node	Operation
FmA24	Final motor assembly operation variant 24
BaA24	Bracket assembly operation variant 24
BaM7	Bracket A machining operation variant 7
BbM13	Bracket B machining operation variant 13
FaA25	Frame assembly operation variant 25
RaA14	WArmature assembly operation variant 14
AaA26	Armature assembly operation variant 26
WsM7	Washer machining operation variant 7
MtaA17	Magnet assembly operation variant 17
CtaA13	Commutator assembly operation variant 13
ClaA13	Coil assembly operation variant 13
FmM4	Frame machining operation variant 4
MhaA26	Magnet housing assembly operation variant 26
MtM8	Magnet machining operation variant 8
MhS2	Magnet housing stamping operation variant 2
CtM3	Commutator machining operation variant 3
StM5	Shaft machining operation variant 5
ClF7	Coil forming operation variant 7
TpS4	Tape shaping operation variant 4

Figure B.25 Motor routing #25

Table B.25 Operations in motor routing #25

Node	Operation
FmA25	Final motor assembly operation variant 25
MbA18	Mainbody B assembly operation variant 18
WtM2	Weight machining operation variant 2
BaA30	Bracket assembly operation variant 30
BaM16	Bracket A machining operation variant 16
BbM6	Bracket B machining operation variant 6
FaA27	Frame assembly operation variant 27
AaA19	Armature assembly operation variant 19
MtaA25	Magnet assembly operation variant 25
CtaA6	Commutator assembly operation variant 6
ClaA17	Coil assembly operation variant 17
FmM10	Frame machining operation variant 10
MhaA23	Magnet housing assembly operation variant 23
MtM5	Magnet machining operation variant 5
MhS10	Magnet housing stamping operation variant 10
CtM5	Commutator machining operation variant 5
StM2	Shaft machining operation variant 2
ClF1	Coil forming operation variant 1
TpS8	Tape shaping operation variant 8

Figure B.26 Motor routing #26

Table B.26 Operations in motor routing #26

Node	Operation
FmA26	Final motor assembly operation variant 26
BaA26	Bracket assembly operation variant 26
BaM8	Bracket A machining operation variant 8
BbM10	Bracket B machining operation variant 10
FaA6	Frame assembly operation variant 6
MtaA7	Magnet assembly operation variant 7
FmM2	Frame machining operation variant 2
MhaA8	Magnet housing assembly operation variant 8
MtM1	Magnet machining operation variant 1
MhS6	Magnet housing stamping operation variant 6
AaA10	Armature assembly operation variant 10
CtaA7	Commutator assembly operation variant 7
ClaA6	Coil assembly operation variant 6
CtM6	Commutator machining operation variant 6
StM1	Shaft machining operation variant 1
ClF8	Coil forming operation variant 8
TpS7	Tape shaping operation variant 7

Figure B.27 Motor routing #27

Table B.27 Operations in motor routing #27

Node	Operation
FmA27	Final motor assembly operation variant 27
BaA24	Bracket assembly operation variant 24
BaM13	Bracket A machining operation variant 13
BbM2	Bracket B machining operation variant 2
FaA7	Frame assembly operation variant 7
MtaA22	Magnet assembly operation variant 22
FmM3	Frame machining operation variant 3
MhaA19	Magnet housing assembly operation variant 19
MtM7	Magnet machining operation variant 7
MhS7	Magnet housing stamping operation variant 7
AaA23	Armature assembly operation variant 23
CtaA19	Commutator assembly operation variant 19
ClaA3	Coil assembly operation variant 3
CtM10	Commutator machining operation variant 10
StM7	Shaft machining operation variant 7
ClF9	Coil forming operation variant 9
TpS5	Tape shaping operation variant 5

Figure B.28 Motor routing #28

Table B.28 Operations in motor routing #28

Node	Operation
FmA28	Final motor assembly operation variant 28
BaA25	Bracket assembly operation variant 25
BaM5	Bracket A machining operation variant 5
BbM16	Bracket B machining operation variant 16
FaA26	Frame assembly operation variant 26
MtaA11	Magnet assembly operation variant 11
FmM8	Frame machining operation variant 8
MhaA24	Magnet housing assembly operation variant 24
MtM5	Magnet machining operation variant 5
MhS2	Magnet housing stamping operation variant 2
AaA25	Armature assembly operation variant 25
CtaA6	Commutator assembly operation variant 6
ClaA8	Coil assembly operation variant 8
CtM5	Commutator machining operation variant 5
StM2	Shaft machining operation variant 2
ClF5	Coil forming operation variant 5
TpS1	Tape shaping operation variant 1

Table B.29 Operations in motor routing #29

Figure B.29 Motor routing #29

Node	Operation
FmA29	Final motor assembly operation variant 29
BaA17	Bracket assembly operation variant 17
BaM20	Bracket A machining operation variant 20
BbM9	Bracket B machining operation variant 9
FaA19	Frame assembly operation variant 19
MtaA1	Magnet assembly operation variant 1
FmM1	Frame machining operation variant 1
MhaA22	Magnet housing assembly operation variant 22
MtM3	Magnet machining operation variant 3
MhS8	Magnet housing stamping operation variant 8
AaA28	Armature assembly operation variant 28
CtaA13	Commutator assembly operation variant 13
ClaA1	Coil assembly operation variant 1
CtM3	Commutator machining operation variant 3
StM5	Shaft machining operation variant 5
ClF7	Coil forming operation variant 7
TpS9	Tape shaping operation variant 9

Table B.30 Operations in motor routing #30

Figure B.30 Motor routing #30

Node	Operation
FmA2	Final motor assembly operation variant 2
MaA4	Mainbody A assembly operation variant 4
RhS2	Rubber holder shaping operation variant 2
WtM1	Weight machining operation variant 1
MbA6	Mainbody B assembly operation variant 6
BaA12	Bracket assembly operation variant 12
FaA11	Frame assembly operation variant 11
RaA11	WArmature assembly operation variant 11
AaA15	Armature assembly operation variant 15
WsM2	Washer machining operation variant 2
MtaA15	Magnet assembly operation variant 15
BaM1	Bracket A machining operation variant 1
BbM3	Bracket B machining operation variant 3
CtaA11	Commutator assembly operation variant 11
ClaA14	Coil assembly operation variant 14
FmM6	Frame machining operation variant 6
MhaA12	Magnet housing assembly operation variant 12
MtM1	Magnet machining operation variant 1
MhS4	Magnet housing stamping operation variant 4
CtM4	Commutator machining operation variant 4
StM3	Shaft machining operation variant 3
ClF5	Coil forming operation variant 5
TpS8	Tape shaping operation variant 8