A FRAMEWORK FOR DEVELOPMENT ARCHITECTURE FOR MODULAR PRODUCTS: CROSS-DOMAIN VARIETY MANAGEMENT PERSPECTIVE

Oh, Kwansuk; Lim, Jong Wook; Cho, Seongwon; Ryu, Junyeol; Hong, Yoo S.

Department of Industrial Engineering & Institute for Industrial Systems Innovation, Seoul National University

ABSTRACT

Variety management is a cross-domain issue in product family design. In the real field, the relationships across the domains are so complex for most of the existing product families that they cannot be easily identified without proper reference architecture. This reference architecture should provide the cross-domain mapping mechanisms in an explicit manner and be able to identify the proper units for management. From this perspective of cross-domain framework, this paper introduces development architecture (DA) to describe the relationships between elements in market, design, and production domains and to give insights for the cross-domain variety management in the product development stage. DA has three parts: (1) the arrangement of elements in each domain, (2) the mapping between elements, and (3) the identification of management sets and key interfaces which are the proper units for variety management. The proposed development architecture framework is applied to the case of front chassis family of modules of an automobile.

Keywords: Product architecture, Product families, Complexity, Variety management

Contact:
Oh, Kwansuk
Seoul National University
Industrial Engineering
Korea, Republic of (South Korea)
oh0421@snu.ac.kr

Cite this article: Oh, K., Lim, J.W., Cho, S., Ryu, J., Hong, Y.S. (2019) 'A Framework for Development Architecture for Modular Products: Cross-Domain Variety Management Perspective', in *Proceedings of the 22nd International Conference on Engineering Design (ICED19),* Delft, The Netherlands, 5-8 August 2019. DOI:10.1017/dsi.2019.299

1 INTRODUCTION

Variety management is a challenging issue for global manufacturing firms (ElMaraghy *et al*., 2013). Manufacturers implement variety management strategies such as product platform (Robertson and Ulrich, 1998), modular design (Baldwin and Clark, 2000), or product family design (Meyer and Utterback, 1993) in order to meet diverse customers' needs while reducing costs. For successful strategies, manufacturers should notice that they need to consider multi-domains including market and production as well as design domains. As shown in Figure 1, product variety is influenced by both external drivers in markets and internal drivers in production (ElMaraghy *et al*., 2013). For example, regional characteristics of markets, dissimilar conditions of segments, and various customer needs drive differentiation in products, while differences in suppliers, factories, and processes in production domain generate additional variety of products.

Cross-domain variety management can be achieved by architecture. Ulrich (1995) defined *product architecture* as "the scheme by which the function of a product is allocated to physical components." He emphasized that the type of relationships between elements in different domains is one of the most important factors for variety generation. In line with his work, how to map relationships between elements in market, design, and production domains is a key consideration on variety management. Hansen and Mortensen (2014) have developed *program architecture* which covers those three domains. Program architecture is a comprehensive planning methodology for product family design, capturing the benefits of alignment of market, product, and production architectures. However, there is a limitation on identifying complex relationships between elements in each domain, in that the alignment cannot be completely achieved in reality. In this vein, two research questions (RQs) of this paper are stated as below.

- RQ1. At the architecture level, how can relationships between elements in market, design, and production domains be described?
- RQ2. How can we manage complexity induced from complex relations between elements in the three domains?

This paper introduces *development architecture* (DA) to describe the relationships between elements in market, design, and production domains and to give insights for the cross-domain variety management in the product development stage. In DA, management sets and key interfaces are derived for efficient management of variety. In section 2, other variety management methodologies in the cross-domain perspective are reviewed. Section 3 describes DA consisting of elements in each domain, their relationships, management sets, and key interfaces. Then, section 4 applies DA to a front chassis component family of an automotive manufacturer, and section 5 discusses applications of DA. Finally, section 6 concludes the paper.

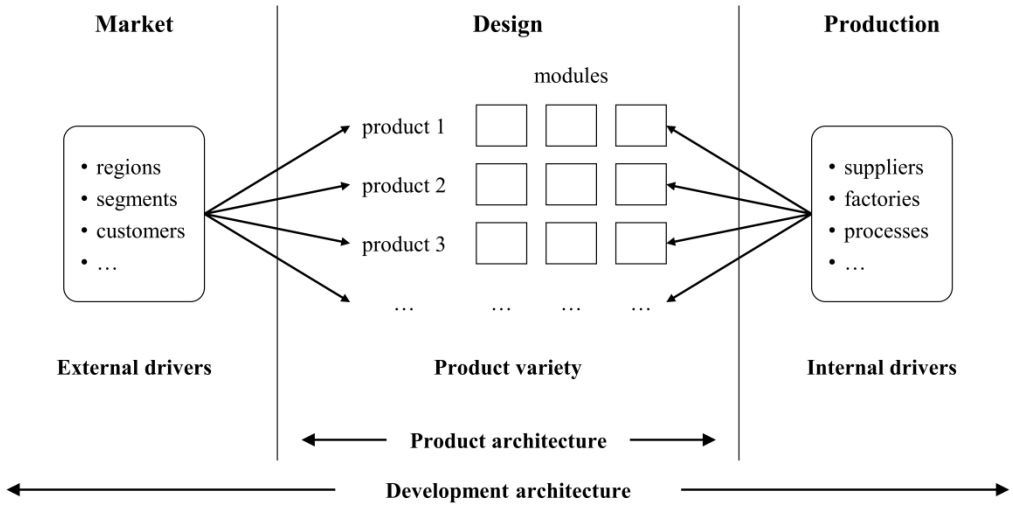

Figure 1. Variety and architecture

https://doi.org/10.1017/dsi.2019.299 Published online by Cambridge University Press

2 LITERATURE REVIEW

Variety management has been studied in a wide range of areas (ElMaraghy *et al*., 2013). This section focuses on variety management methodologies based on the cross-domain and architectural approaches. Two major streams of the issue are (1) modular product family design and (2) product family architecture.

2.1 Modular product family design

There are a few papers dealing with product family design which is based on modular product architecture. Erixon (1998) developed a framework for modularization called modular function deployment (MFD) to identify an appropriate level of modules for variety management. He found a set of module drivers in the product lifecycle such as design, production, quality, and purchasing. Gonzalez-Zugasti and Otto (2000) proposed an optimization method for obtaining design solutions for products in a family. Jiao *et al*. (2007) reviewed product family design researches related to product platform concept. They summarized the issues in a holistic view consisting of customer, functional, physical, process, and logistics domains. Simpson *et al*. (2012) proposed an approach to integrate product family design tools into a framework to translate customer requirements into commonality specifications. After that, Otto *et al*. (2016) established a general framework for modular product family design from market analysis to architecture roadmap design while reviewing related researches. Another approach for modular product family design called Integrated PKT-Approach is developed by Institute of Product Development and Mechanical Engineering Design (PKT; Krause and Eilmus, 2011; Krause and Ripperda, 2013). Motivation of this approach is to reduce the complexity of internal variety which is affected by external variety. The approach combines product-oriented and process-oriented views and supports decisions on product variety by providing the analysis and visualization tools, for example, variety allocation model (VAM) suggested by Blees *et al*. (2010). VAM is a tool for allocation between attributes, functions, working principles, and components. Recently, Ripperda and Krause (2017) proposed a method for quantification of variety-induced complexity throughout the product lifecycle, and Hanna *et al*. (2018) constructed a data model for consistency of product family information. Many papers in this field of product family design have recognized the need for the consideration of cross-domain variety management, however from the architectural viewpoint, the researches have focused mainly on the product architecture covering functions and modules within the design domain rather than dealing with cross-domain architecture relating market, design, and production domains.

2.2 Product family architecture (cross-domain viewpoint)

After Ulrich (1995) claimed that the key factor of an architecture is the mapping between elements in different domains, several concepts of an architecture have been proposed in the product family design field. Tseng and Jiao (1998) defined product family architecture (PFA) as "the underlying architecture within which various products can be derived from basic product designs to satisfy a spectrum of customer needs." They included functional, behavioral, and structural views into the architecture, and connected the views for generation of variety of objects in a product family. For practical improvement, Jiao *et al*. (2000) constructed a data structure, named generic bill-of-materials-and-operations (BOMO), integrating customer order, engineering, and operations data for comprehensive management. Du *et al*. (2001) introduced architecture of product family (APF) for logical configurations of products. APF is also constructed based on the multiple views of marketing, engineering, and manufacturing.

Harlou (2006) systematically summarized vocabularies for architectures such as design units, standard designs, interfaces, and modules. Then, he proposed two supporting tools for architectures of a product family, named generic organ diagram and product family master plan (PFMP). PFMP reflects three aspects of modelling a product family which are customer, engineering, and part views. From the three viewpoints, Mortensen *et al*. (2011) modelled market, product, and production architectures respectively. Integrating the previous works, Hansen and Mortensen (2014) developed program architecture in which the three architectures are aligned for multi-level development roadmaps of product families. In their work, the term "alignment" is a key characteristic distinguishing program architecture from product architecture. Bonev *et al*. (2013) established product requirement development model for reflecting customer needs to multi-layered architecture by combining PFMP with matrix-based methods. Other tools such as interface diagram (Bruun *et al*., 2015) and architecture

https://doi.org/10.1017/dsi.2019.299 Published online by Cambridge University Press

mapping and evaluation (AME; Mortensen *et al*., 2016) were continuously advanced to support the overall framework for product family architecture design.

2.3 Summary

Cross-domain variety management has been mentioned widely in the fields of product family design and product family architecture. Among those, the program architecture (Hansen and Mortensen, 2014) needs special attention because it has raised explicitly the cross-domain issues of product family design and has emphasized importance of aligning the three architectures in market, design, and production domains. In the real field, however, the relationships across the domains are so complex for most of the existing product families that they cannot be easily identified without proper reference architecture. This reference architecture should provide the cross-domain mapping mechanisms in an explicit manner and be able to identify the proper units for management. From this perspective, we first introduce the development architecture (DA) for cross-domain variety management, then provide a methodology for defining the management sets and identifying the key interfaces between them.

3 DEVELOPMENT ARCHITECTURE

Product architecture is precisely defined as the arrangement of elements, the mapping from functional elements to physical components, the classification of components to modules, and the specification of interfaces (Ulrich, 1995). From the same perspective of cross-domain framework, *development architecture* (DA) is defined as the arrangement of attributes, modules, and facilities in market, design, and production domains, respectively, the mapping from attributes to modules to facilities, the classification of elements to management sets, and the identification of key interfaces. Figure 2 shows the overall framework for DA.

3.1 Arrangement of elements

As shown in Figure 2, DA has the three domains: market, design, and production, and each domain has base units: attributes, modules, and facilities, respectively. An *attribute* (A), which is a base unit of market domain, represents a characteristic of a product with which customers distinguish between products (Robertson and Ulrich, 1998). In the automotive industry, for example, drive type and weather type can be attributes because a firm develops differentiating products by sales regions based on those

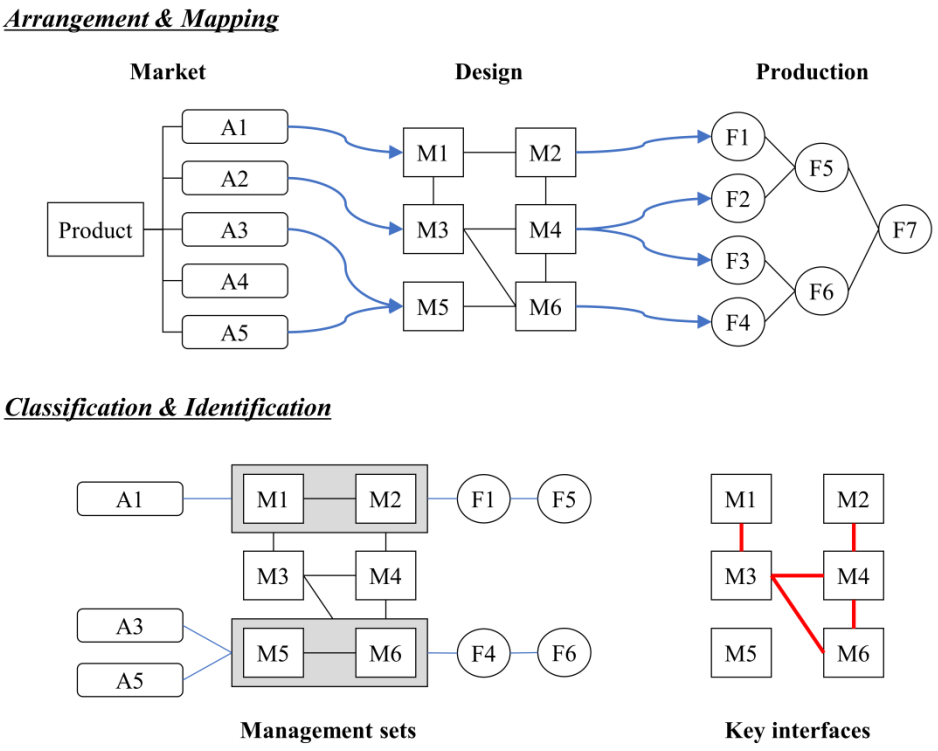

Figure 2. Development architecture

https://doi.org/10.1017/dsi.2019.299 Published online by Cambridge University Press

attributes. Vehicle width and wheel size can also be attributes as customers recognize differences between products according to the values they appreciate. An attribute is usually specified by a deterministic value such as '15 inch' or '16 inch' for wheel size for example. A manufacturer can generate a product by configurations of different levels of attributes.

In design domain, a *module* (M) is regarded as a base unit. The modular product architecture, on which this paper focuses, has advantages over the integral product architecture in generation of products by replacing modules (Ulrich, 1995; Baldwin and Clark, 2000). Functional elements in the modular product architecture are encapsulated into physical chunks, or modules, so that the modules can be individually developed and connected with other modules through proper interfaces between them. In DA, a product is first configured by attribute levels in market domain, then physically composed into module variants in design domain. Thus, a module variant has unique specifications which correspond to the related attribute levels.

In the production domain of DA, a *facility* (F) is a base unit. A facility is regarded as a place where manufacturing and assembly processes are executed to produce modules and final products. Facilities include suppliers, module assembly lines, final assembly lines, etc. The facilities may be geographically dispersed because manufacturers generally outsource modules to various suppliers. In addition, a single supplier may produce multiple module variants, while a number of variants of a module may be sometimes produced by different suppliers. In this situation, since manufacturing and assembly processes of a facility are usually shared across module variants and products, a firm's sourcing decision is extremely important for variety management, considering the cost savings by economies of scale as well as scope.

3.2 Mapping from attributes to modules to facilities

After the arrangement of elements, the allocation of attributes to modules, and to facilities is progressed. The mapping from an attribute to a module means that the module needs variations by differentiating levels of the attribute. For example, a steering gear module of an automobile is differentiated by drive type and engine type, so the mapping between them should exist. The mapping of DA should be decided carefully after discussing with designers. Subsequently, the allocation from modules to facilities is decided. Although the mapping can be a rough allocation plan, this is the important step for variety management. For example, if the left-hand drive steering gear is planned to be outsourced to supplier *A*, and the right-hand drive one to supplier *B*, then there exist two relationships, i.e., from the single steering gear module to the two respective facilities.

Types of the relationships between elements vary according to the allocation plan such as one-to-one, many-to-one, one-to-many, and many-to-many. For variety management, the one-to-one mapping is more efficient than others because there is no complex relationships. However, the mappings in reality cannot be completely one-to-one due to many practical situations the manufacturers are faced with. Thus, classification of the relationships between elements is needed to constitute management sets with which efficiently manage several modules together.

3.3 Classification of elements to management sets and identification of key interfaces

In this step, we classify the complex relationships into management sets and identify key interfaces that need to be standardized. A *management set* is a set of modules related to the same attribute and connected through interfaces. The classification of modules related to attributes and facilities is determined by the following criteria (CRs).

- CR1. Modules in a set are interacted through interfaces within the product structure.
- CR2. (In the attribute-module mapping) Modules have the same relationships with attributes, or one module covers all the relationships with attributes of the other module.
- CR3. (In the module-facility mapping) Modules share one or more facilities.

Figure 3 is an example of the classification. The results of mappings are represented by the cross-domain mapping matrices as shown on the top of Figure 3. A mark 'X' means that there is a relationship between a row element and a column element. After deriving the mapping matrices and the product structure, the matrices are classified by the criteria, and the management sets and the key interfaces are derived at the bottom of Figure 3. For example, M1 and M3 are connected within the product structure (CR1), they

https://doi.org/10.1017/dsi.2019.299 Published online by Cambridge University Press

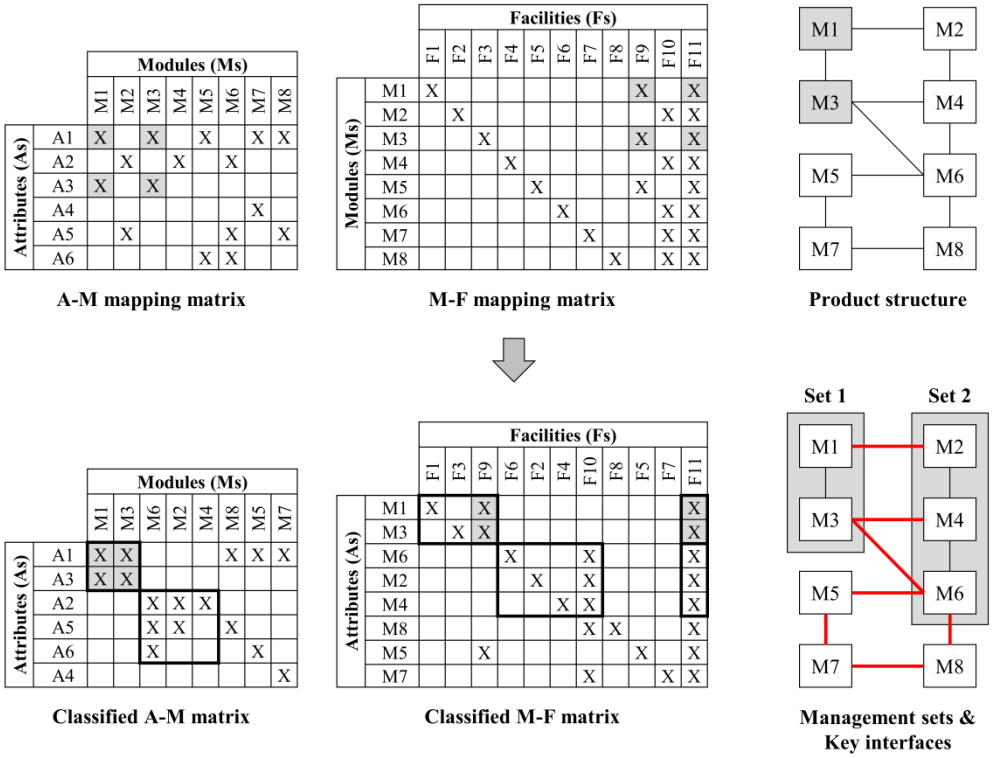

Figure 3. Classification of the mapping

are related to the same attributes A1 and A3 (CR2), and they share assembly processes taking place in F9 and F11 (CR3). Thus, M1 and M3 is combined as a management set.

Another example of management sets is the case of M6, M2, and M4 which satisfies the criteria as well. Management sets generate instances when specific attribute levels, module variants, and production processes are planned. The instances will be shown in the case study.

A *key interface* is an interface that should be standardized or stabilized by design rules (Mortensen and Løkkegaard, 2017). By determining the design rules, the key interface blocks the unexpected changes propagated from other modules. When a manufacturer develops variety of products, the key interface enables diverse configurations of module variants. Løkkegaard *et al.* (2018) argued that establishing design rules is the critical step for managing product and manufacturing architectures. They defined business critical design rules (BCDRs) between products and manufacturing lines by mapping the architectures in design and production domains. From the same viewpoint, the key interface in this paper considers the market domain as well as the design and production domains, and is derived by configuring the attributes of a product.

When a manufacturer generates products by replacing modules, a number of combinations of module variants can exist. However, not all interfaces should be standardized because the configurations of attributes are aligned for modules in a management set. A module variant in the same set is always connected with other module variants related to the same attribute levels. For example, a brake assembly module for *'15-inch wheel size*' of an automobile is always configured with '*15-inch wheel size*' wheel, not with '16 inch' or '17 inch'. Thus, as shown in bold lines at the bottom right side of Figure 3, we can set the interfaces outside of the management sets as key interfaces. Module designers need to discuss the design rules in advance before products are developed.

4 ARCHITECTING WITH A FRONT CHASSIS CASE

4.1 Case description

In this section, the proposed development architecture framework is applied to the case of front chassis family of modules of an automobile. A front chassis family is a part of an automotive platform of which the constituent modules need some variations to satisfy global markets and customers. The front chassis is composed of nine modules as shown in Figure 4: brake assembly, drive shaft, axle assembly, cross member, lower arm, stabilizer bar, steering gear, strut assembly, and roll stopper. The

https://doi.org/10.1017/dsi.2019.299 Published online by Cambridge University Press

manufacturer develops the front chassis family by replacing modules. The nine modules are produced by different factories and suppliers, and the manufacturer assembles them in a single assembly line. Since there are a number of products and module variants involved, it is necessary to establish DA to manage the front chassis family in an efficient and effective manner.

4.2 Development architecture

The first step of architecting is the arrangement of base units. Since the front chassis is already divided into nine modules, this step starts from the arrangement of market attributes which are related to the modules. After discussion with designers, eleven attributes related to modules are selected: drive type, weather type, vehicle width, body type, engine type, transmission type, MDPS type, disc size, brake performance, shock absorber performance, and suspension type. The attributes are arranged in the rows of the left matrix in Figure 5. For facilities, there already have been seven module factories and two suppliers producing nine modules respectively, and one assembly factory for assembling all modules to a final product. The facilities are arranged in the columns of the right matrix in Figure 5. The relationships between attributes, modules, and facilities are 'X' marked on the matrices.

In the next step, we identify the management sets based on the criteria stated in the previous section. After discussion with designers, the total of three management sets are derived: brake assembly & axle assembly (management set 1), cross member, steering gear, & roll stopper (management set 2), and strut assembly & stabilizer bar (management set 3). The classified matrices are described in Figure 6. For management set 1, the brake assembly and axle assembly modules are related to the attributes of disc size and brake performance. In addition, the two modules share the assembly processes in the same assembly factory. Management sets for the horizontally symmetric product structure are represented in Figure 7. Bold lines in the figure represent the key interfaces connecting beyond the boundaries of the management sets. Specific design rules and parametric values of each interface are not treated in this case study. Instead, we derive the instances of management sets based on the product family configuration plan. When attribute levels, module variants, and production processes are planned,

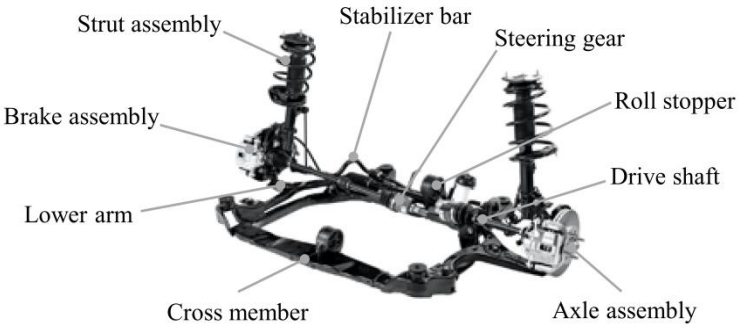

Figure 4. Front chassis

		Modules (Ms)								
		Brake assy.	Drive shaft	Axle assy.	Cross member	Lower arm	Stabilizer bar	Steering gear	Strut assy.	Roll stopper
Attributes (As)	Drive type				X			X		
	Weather type				X					
	Vehicle Width					X				
	Body type						X		X	
	Engine type		X		X		X	X	X	X
	T/A type		X							
	MDPS type				X			X		
	Disc size	X		X						
	Brake performance	X		X						
	S/A performance								X	
	Suspension type		X			X			X	

		Facilities (Fs)										
		Factory 1-1	Factory 1-2	Factory 1-3	Factory 1-4	Factory 1-5	Factory 2	Factory 3	Factory 4	Supplier 1	Supplier 2	Assembly factory
Modules (Ms)	Brake assy.						X					X
	Drive shaft									X		X
	Axle assy.	X						X				X
	Cross member		X									X
	Lower arm			X								X
	Stabilizer bar				X							X
	Steering gear								X			X
	Strut assy.					X						X
	Roll stopper										X	X

Figure 5. Mapping result of the case

https://doi.org/10.1017/dsi.2019.299 Published online by Cambridge University Press

Attributes (As)	Modules (Ms): Brake assy.	Axle assy.	Cross member	Steering gear	Roll stopper	Strut assy.	Stabilizer bar	Lower arm	Drive shaft
Disc size	X	X							
Brake performance	X	X							
Drive type			X	X					
Weather type			X						
MDPS type			X	X					
Engine type			X	X	X	X	X		X
Body type						X	X		
S/A performance						X			
Suspension type						X		X	X
Vehicle Width								X	
T/A type									X

Modules (Ms)	Facilities (Fs): Factory 2	Factory 1-1	Factory 3	Factory 1-2	Factory 4	Supplier 2	Factory 1-5	Factory 1-4	Factory 1-3	Supplier 1	Assembly factory
Brake assy.	X										X
Axle assy.		X	X								X
Cross member				X							X
Steering gear					X						X
Roll stopper						X					X
Strut assy.							X				X
Stabilizer bar								X			X
Lower arm									X		X
Drive shaft										X	X

Figure 6. Classification result of the case

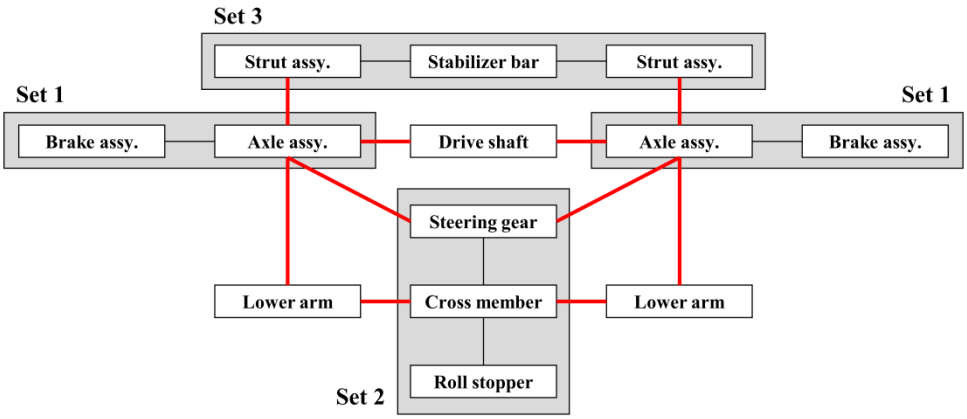

Figure 7. Management sets and key interfaces of the case

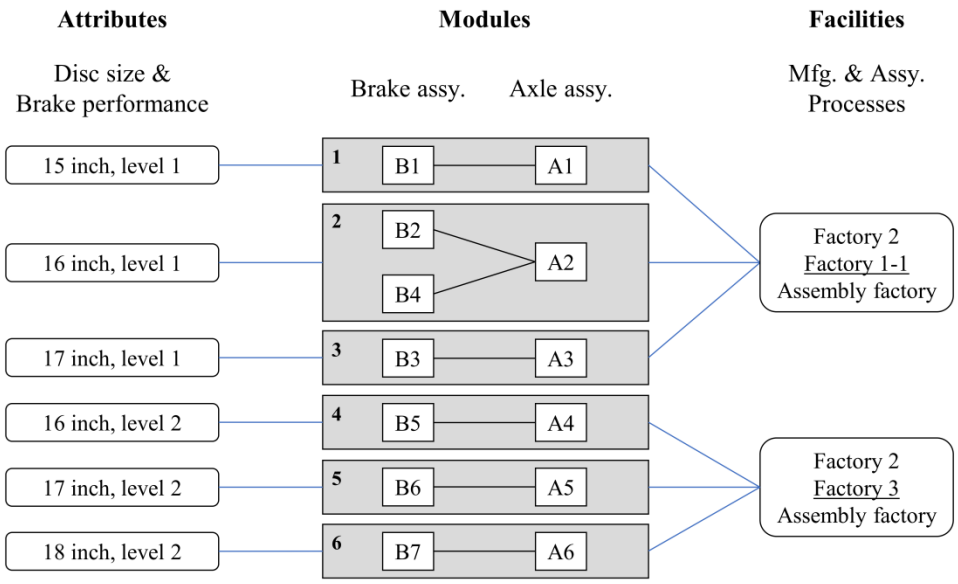

Figure 8. Instances of management set 1

configurations of attributes and modules of a product can be identified. From the configurations, six instances of management set 1 are derived as shown in Figure 8. Each instance has own attribute levels, and instance 1~3 and 4~6 can be separated by production facilities. An instance is considered as a management unit for variety management of a product family.

https://doi.org/10.1017/dsi.2019.299 Published online by Cambridge University Press

5 DISCUSSION

The development architecture (DA) focuses on the relationships between elements in different domains. The type of relationships is varied by various situations that a manufacturer is faced with. In this context, DA classifies the relationships into management sets which are the groups of cross-domain elements, and identifies key interfaces that should be standardized. In this section, we discuss three possible application areas of DA that are variety planning, variant lifecycle management, and change management.

First, DA can be applied to the decisions on variety planning. Variety planning of a product family consists of differentiation plan and commonality plan (Robertson and Ulrich, 1998). Differentiation plan decides specific attribute levels of a product, while commonality plan determines module combinations of a product, considering a process sharing in production domain. From the cross-domain perspective of variety planning, DA can efficiently inform the planner of which attribute levels, module variants, and production processes are linked one another.

Another application area is the field of variant lifecycle management. When a manufacturer plans to develop a product family, products in a family are on different timelines, although the products share module variants and production processes. Thus, the timeline of a module variant should be planned, in terms of development, production, upgrade, and discontinuation, in accordance with the timelines of the products. The management set of DA can give us a direction for variant lifecycle management since the module variants in the same management set should have similar lifecycles.

Change management is one of the most important issues in variety management. Change management is to predict the impacts of and to cope with the propagation of change from one part of a product to others. A change is propagated to all products, not just a single product in the product family design, hence a methodology for predicting and managing the impact of change is needed. DA can give insights for change management by the allocation of elements, and the standardization of key interfaces can help prevent the propagation of change.

6 CONCLUSION

This paper defines development architecture (DA) from the perspective of cross-domain variety management. DA establishes the relationships between elements in market, design, and production domains, and identifies management sets and key interfaces which are the units of variety management. Compared to other methodologies for variety management, this paper focuses on managing the complexity induced from complex relationships between the elements across those domains. In the case study, DA is applied to a front chassis family. On the practical side, further studies to construct a more concrete methodology or framework are needed since DA considers only the basic relationships yet. Thus, a more generalized framework for variety management and its supporting tools such as costing or visualization models would be needed in the future work. On the theoretical side, decision support models for variety planning, variant lifecycle management, and product family change management should be developed following by the concept of development architecture proposed in this paper.

REFERENCES

Baldwin, C.Y. and Clark, K.B. (2000), *Design rules: The power of modularity*, Vol. 1, MIT press, Cambridge, MA.

Blees, C, Kipp, T., Beckmann, G. and Krause D. (2010), "Development of modular product families: Integration of design for variety and modularization", *International NordDesign Conference*, Göteborg, Sweden, 25.-27.08.2010, The Design Society, pp. 159–170.

Bonev, M., Wörösch, M., Hauksdóttir, D. and Hvam, L. (2013), "Extending product modeling methods for integrated product development", *International Conference on Engineering Design*, Seoul, Korea, 19.-22.08.2013, The Design Society, Vol. 4, pp. 219–228.

Bruun, H.P.L., Mortensen, N.H., Harlou, U., Wörösch, M. and Proschowsky, M. (2015), "PLM system support for modular product development", *Computers in Industry*, Vol. 67, pp. 97–111.

Du, X., Jiao, J. and Tseng, M.M. (2001), "Architecture of product family: fundamentals and methodology", *Concurrent Engineering*, Vol. 9 No. 4, pp. 309–325.

ElMaraghy, H.A., Schuh, G., ElMaraghy, W.H., Piller, F., Schönsleben, P., Tseng, M. and Bernard, A. (2013), "Product variety management", *CIRP Annals-Manufacturing Technology*, Vol. 62 No.2, pp. 629–652.

Erixon, G. (1998), "Modular function deployment: A method for product modularisation", *Designation*, Royal Inst. of Technology.

https://doi.org/10.1017/dsi.2019.299 Published online by Cambridge University Press

Gonzalez-Zugasti, J.P. and Otto, K.N. (2000), "Modular platform-based product family design", *Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Baltimore, Maryland, 10.-13.09.2000, American Society of Mechanical Engineers, New York, Vol. 2, pp. 677–688.
Hanna, M., Schwede, L.N. and Krause, D. (2018), "Model-based consistency for design for variety and modularization", *International DSM Conference*, Trieste, Italy, 15.-17.10.2018, The Design Society, pp. 239–248.
Hansen, C.L. and Mortensen, N.H. (2014), "Proactive identification of scalable program architecture: How to achieve a quantum-leap in time-to-market", *International Journal of Industrial Engineering*, Vol. 21 No.2, pp. 74–85.
Harlou, U. (2006), "*Developing product families based on architectures*", *Desingation*, Technical University of Denmark, Copenhagen, Denmark.
Jiao, J., Tseng, M.M., Ma, Q. and Zou, Y. (2000), "Generic bill-of-materials-and-operations for high-variety production management", *Concurrent Engineering*, Vol. 8 No. 4, pp. 297–321.
Jiao, J., Simpson, T.W. and Siddique, Z. (2007), "Product family design and platform-based product development: a state-of-the-art review", *Journal of intelligent Manufacturing*, Vol. 18 No. 1, pp. 5–29.
Krause, D. and Eilmus, S. (2011), "A methodical approach for developing modular product families", *International Conference on Engineering Design*, Copenhagen, Denmark, 15.-19.08. 2011, The Design Society, Vol. 4, pp. 299–308.
Krause, D. and Ripperda, S. (2013), "An assessment of methodical approaches to support the development of modular product families", *International Conference on Engineering Design*, Seoul, Korea, 19.-22.08. 2013, The Design Society, Vol. 4, pp. 31–40.
Løkkegaard, M., Mortensen, N.H. and Hvam, L. (2018), "Using business critical design rules to frame new architecture introduction in multi-architecture portfolios", *International Journal of Production Research*, Vol. 56 No.24, pp. 7313–7329.
Meyer, M. and Utterback, J. (1993), "The product family and the dynamics of core capability", *Sloan Management Review*, Vol. 34 No.3, pp. 29–47.
Mortensen, N.H., Hansen, C.L., Hvam, L. and Andreasen, M.M. (2011), "Proactive modeling of market, product and production architectures", *International Conference on Engineering Design*, Copenhagen, Denmark, 15.-19.08. 2011, The Design Society, Vol. 4, pp. 133.
Mortensen, N.H., Hansen, C.L., Løkkegaard, M. and Hvam, L. (2016), "Assessing the cost saving potential of shared product architectures", *Concurrent Engineering*, Vol. 24 No. 2, pp. 153–163.
Mortensen, N.H. and Løkkegaard, M. (2017), "Good product line architecture design principles", *International Conference on Engineering Design, Vancouver*, Canada, 21.-25.08. 2017, The Design Society, Vol. 3, pp. 141–150.
Otto, K., Hölttä-Otto, K., Simpson, T.W., Krause, D., Ripperda, S. and Moon, S.K. (2016), "Global views on modular design research: linking alternative methods to support modular product family concept development", *Journal of Mechanical Design*, Vol. 138 No. 7, p. 071101.
Ripperda, S. and Krause, D. (2017), "Cost effects of modular product family structures: Methods and quantification of impacts to support decision making", *Journal of Mechanical Design*, Vol. 139 No. 2, p. 021103.
Robertson, D. and Ulrich, K. (1998), "Planning for product platforms", *Sloan Management Review*, Vol. 39 No. 4, pp. 19.
Simpson, T.W., Bobuk, A., Slingerland, L.A., Brennan, S., Logan, D. and Reichard, K. (2012), "From user requirements to commonality specifications: an integrated approach to product family design", *Research in Engineering Design*, Vol. 23 No. 2, pp. 141–153.
Tseng, M.M. and Jiao, J. (1998), "Design for mass customization by developing product family architecture", *Design Engineering Technical Conferences*, Atlanta, Georgia, 13.-16.09.1998, American Society of Mechanical Engineers, New York, DETC98/DTM-5717.
Ulrich, K. (1995), "The role of product architecture in the manufacturing firm", *Research policy*, Vol. 24 No. 3, pp. 419–440.

ACKNOWLEDGMENTS

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2017R1E1A1A03070846).

https://doi.org/10.1017/dsi.2019.299 Published online by Cambridge University Press