공학박사 학위논문

Architectural Approaches to Strategic Product Development

전략적 제품개발을 위한

아키텍처 관점의 접근법

2014 년 2 월

서울대학교 대학원

기술경영경제정책

강 길 모

Abstract

Architectural Approaches to Strategic Product Development

Kilmo Kang
Technology Management Economics and Policy Program
The Graduate School
Seoul National University

While traditional design research has concentrated on creativity from a clean sheet, however in practice many design projects have been conducted by the modification or incremental development of existing systems to meet new requirements and regulations. Indeed, *Ab initio* designing is rare, while many new product developments proceed by modifying existing products. Radical design, which begins from white paper, requires new knowledge that carries higher uncertainty and an increased risk of market failure, compared with existing knowledge. Although many enterprises expect more success from radical innovations, most new products only improve or modify existing products. Therefore, minimizing novelty to reduce risk and cost, by using tried and tested solutions and carried-over components, is a key objective. In many industries, more formal procedures for specifying tight and complex requirements are changing the nature of decision making in design processes.

The present thesis defines incremental design as a process of modifying or redesigning an existing system while carrying over core competencies in order to meet the required incremental changes and propose the methodologies to established effective strategies for the incremental design. In order to success in incremental product development, it is primary to comprehensively understand the existing product's architecture. On the foundation of the understanding, determination of the design targets and effective realization on physical domain should be systemically conducted. The product architecture is defined as the scheme by which the function of a product is allocated to physical components. Therefore, in the incremental design, the existing product's architecture could be a design constraint for a new product. For determining design targets on the early stage of product development process, the proposed methodology figures out the interrelationships among functional elements, which specify the product's tasks, and based on this, determines the consistent set of specifications that make a product satisfy new requirements. The determined specifications are implemented or realized with physical components on the physical domain. When the existing system incorporates new components in incremental design, reduced changes should be necessarily accompanied. Therefore, efficient rearrangement of the existing components with incorporating new components should be a key design strategy in incremental design.

In order to determine a consistent set of design targets in incremental design, the proposed methodology defines the product architecture with specifications on the functional design domain and identifies the specifications that

makes customers' utility maximize; the design targets are specified with specifications. The methodology was practically developed based on new vehicle planning project, because it traditionally has been conducted in incremental manner, which relies solely on qualitative benchmarking analysis and intuitive human decisions. It has tried to capture the interplay between the important factors in preliminary vehicle design such as functional product architecture (design feasibility constraints), market demands, and economic conditions. The main contribution of the proposed research could read as showing how design information embedded in real data can be utilized in vehicle planning and determine a consistent set of design targets by coordinating those design information on moderate level.

The determined design targets are implemented with physical components. When a new product is developed as based on an existing system but with new components, changing not only the components but also the entire architecture on physical domain is unavoidable. Therefore, this thesis proposes a methodology to re-architect an existing system that has modular architecture when new technologies are to be infused via a set of new components. The proposed method explicitly recognizes the existing system, as the foundation of the new system, focuses on the transformation of the existing architecture into the optimal architecture of the new system. Vast amount of prior research on designing modular systems or building product platforms have proposed numerous methodologies to determine the optimal architecture for developing new products, implicitly acknowledging the existence of the previous design. Although it is imperative to determine the goal of the new

architecture, the paths of transformation, from the existing to the optimal architecture for a new system, can be exceedingly varied; furthermore, the optimal architecture itself should depend on the transformation path selected to meet the new requirements. Therefore, the method proposed in this paper models the transformation of an existing architecture by reflecting required changes. The present study determined the optimal architecture by consideration of the relevant transformation characteristics.

Keywords: Incremental design; product architecture; design strategy; design target; design constraint; re-architecting; modularity

Student Number: 2007-20876

Contents

List of Tables

List of Figures

Chapter 1. Introduction

While traditional design research has concentrated on creativity from a clean sheet, however in practice many design projects have been conducted by the modification or incremental development of existing systems to meet new requirements and regulations. To systemically establish the incremental design strategies, it is imperative to definitely understand and configure the architecture of the existing systems. On the foundation of the investigation, this paper proposes several design strategies.

1.1. Incremental design

Design can be defined as *an interplay between what to achieve and how to achieve it* (Suh 2001). In new product development, design activity should clarify the design targets (what to achieve through the new product) and effectively realize the design targets through a clear description of how to achieve it. Axiomatic design (Suh 2001) suggest that once designers understand the customer's needs, this understanding should be transformed into a minimum set of specifications that adequately describe "what to achieve" to satisfy the

customers' needs. Thereafter, the minimum set of specifications are realized through physical components that embrace the customers' requirements as much as possible. These interplays can be applied to any scope of design problem on new product development. Creative thinking in design as the imagining, very early in the design process, of a new product that meets the customer's requirements in innovative ways has conducted a primary role in new product development.

While traditional design research has focused on creativity in the early phased of design and on creativity in very open-ended design tasks, however, in practice many design projects concern the modification or incremental design of existing systems to meet new needs and restrictions. Indeed, as design practice has become complicated in complex projects, *ab initio* designing is rare. Rather than beginning from a clean paper, many design projects proceed by modifying existing products. The novelty of new product is measurement of its innovativeness. The novelty can be differentiated according to the kind of knowledge where the novelty is based; exploitative innovations are based on existing knowledge, and explorative innovations are based on entirely new knowledge (Benner and Tushman 2003). Radical design, which begins from white paper, requires new knowledge that carries higher uncertainty and an increased risk of market failure, compared with existing knowledge. Indeed, although many enterprises expect more success from radical innovations, most new products only improve or modify existing products (Cooper 2001). Therefore, minimizing novelty to reduce risk and cost, by using tried and

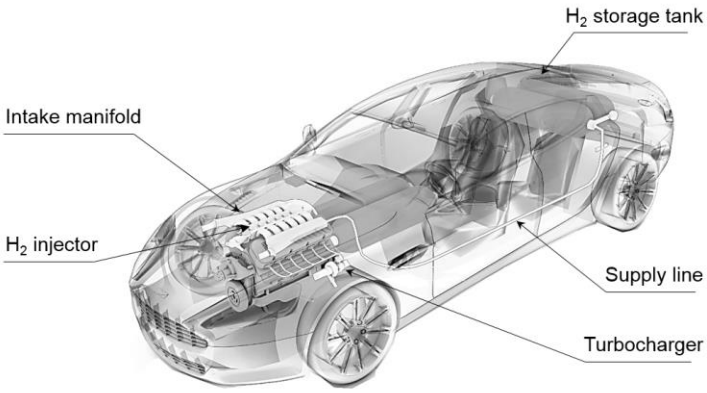


Figure 1-1. Hybrid Hydrogen Fuelled Internal Combustion Vehicle (AlsetGlobal 2013)

tested solutions and carried-over components, is a key objective. In many industries, more formal procedures for specifying tight and complex requirements are changing the nature of decision making in design processes (Eckert et al. 2012).

In this research, *incremental design* is defined as *a process of modifying or redesigning an existing system while carrying over core competencies in order to meet the required incremental changes*. It should go without saying that by this definition, designing an entirely new system without carrying over existing system does not constitute incremental design. Conversely, simply modifying ore redesigning a previous design, without incorporation of new technologies or innovations, cannot be considered incremental design.

As an appropriate example of incremental design, very recently Aston Martin's Rapide S race car adapted hybrid hydrogen system with minimal

modifications to the internal combustion engine. The hybrid hydrogen system developed by Alset Global Company has been incrementally designed as a transitional technology that harnesses the advantages of hydrogen and decades of development of the internal combustion engine while avoiding the current obstacles to the widespread introduction of hydrogen-powered vehicles (AlsetGlobal 2013). According to Figure 1-1. Hybrid Hydrogen Fuelled Internal Combustion Vehicle, engine modifications are limited to the inclusion of turbochargers and associated intercooler, along with compression ratio reductions, special valve seats and injectors.

In the nature of incremental design, existing system's characteristics such as interrelations between specifications, module composition, and the interdependencies between modules (components) are carried over to the new system without alteration. This reapplication of core characteristics reduces both design costs and risks. Specifically, complex-system design incurs reliability and feasibility risks (Wyatt et al. 2009); reusing ore slightly modifying existing system that have been tested and verified through one or more product generations can alleviate or even remove such risks (Eckert et al. 2012). Considering that modules define the boundary that creates dependency between components in higher and lower modules, companies establish practical information-flow systems while designing products to reflect the product modules. Henderson and Clark (1990) showed that architectural innovation that changes a product's core design concepts nullifies the usefulness of an established firm's architectural knowledge. Moreover, because module interfaces and corresponding team interactions are significantly aligned (Sosa et al. 2004),

breaking the boundary between modules can cause misalignment, thus, can hinder the efficient transfer of design information. In global product development, it is important to retain the boundaries of the existing modules in that the production or development tasks by offshore sites are composed based on that module boundaries (Tripathy and Eppinger 2011). Certainly, the carrying over of existing system is integral to the efficiency of incremental-design-based new-system design.

The existing system acts as design constraints in incrementally designing new system. As the existing system is baseline for the new system, the carried-over characteristics should define boundary where new system can change. This boundary forms design space where all potential design solutions exist, and incremental design explore and exploit the best solution within the design space. The design constraints from the existing system can be presented as various forms according to design domains where the incremental design problem is issued. The detailed concept of the design domain and the related constraint forms of incremental design are following the next section.

1.2. Role of Product Architecture in Incremental design

In this research, three *design domains* are addressed to systematize the thought process involved an interplay between "what to achieve" and "how to achieve" in incremental design: *customer domain*; *functional domain*; *physical domain*. The customer domain is characterized by the attributes that the customer is looking for in a new product. The key attributes of the product are specified in

terms of functional requirements (elements) and their interrelationships. Finally, in the physical domain, the specified functional requirements are realized with physical components or modules. Interfaces between physical components are also defined. Note that the three design domains are defines based on the concept of domains in axiomatic design; according to axiomatic design (Suh 2001), the world of design is made up of four domains: the customer domain, the functional domain, the physical domain, and the process domain. The process domain is characterized by process variables to produce the product specified in terms of physical components.

Product architecture is the scheme by which the function of a product is allocated to physical components. Product architecture is more precisely defined as: (1) the arrangement of functional elements; (2) the mapping from functional elements to physical components; (3) the specification of the interfaces among interacting physical components (Ulrich 1995). First, the arrangement of functional elements and their interrelations consist of function structure (Pahl et al. 2007). The functional elements describes what the product achieve for the potential users. Functional elements are sometimes called *functional requirements* (Suh 2001) or *functives* (Fowler 1990), and the function diagram has been variously called a *function structure* (Pahl et al. 2007), a *functional description* and a *schematic description* (Ulrich and Seering 1989). Herein, the interrelations between functional elements can involve the exchange of signals, materials, forces and energy. The second part of the product architecture is the mapping from functional elements to physical components. For clarity, a physical component is defined as a separable physical part. A set

of the physical components can compose a module that is independently mapped to a function element. Also, individual functional elements can be mapped to several components or modules. How to compose physical components to a group and map it to functional elements affects effectiveness and efficiency of design processes (Baldwin and Clark 2000). The third part of the product architecture is the specification of the interfaces among interacting physical components. The mapped physical components are connected each other by some physical interface. The interfaces may involve geometric connections between two components or may involve non-contact interactions such as communicated links. Therefore, how to specify and design the interfaces between physical components should affect the whole characteristics of the new product.

Product architecture penetrates three design domains. The design information flow different domains through the product architecture and organically cross-refer each other. Therefore, systemically establishing product architecture to integrate design information in consistent manner is imperative for effective/efficient design activity. Figure 1-2 represents product architecture penetrating design domains in incremental design. Functional elements, which specify customer requirements defined on customer domain, are interconnected each other on the functional domain. The functional elements are mapped to components on the physical domain. Physical components interact

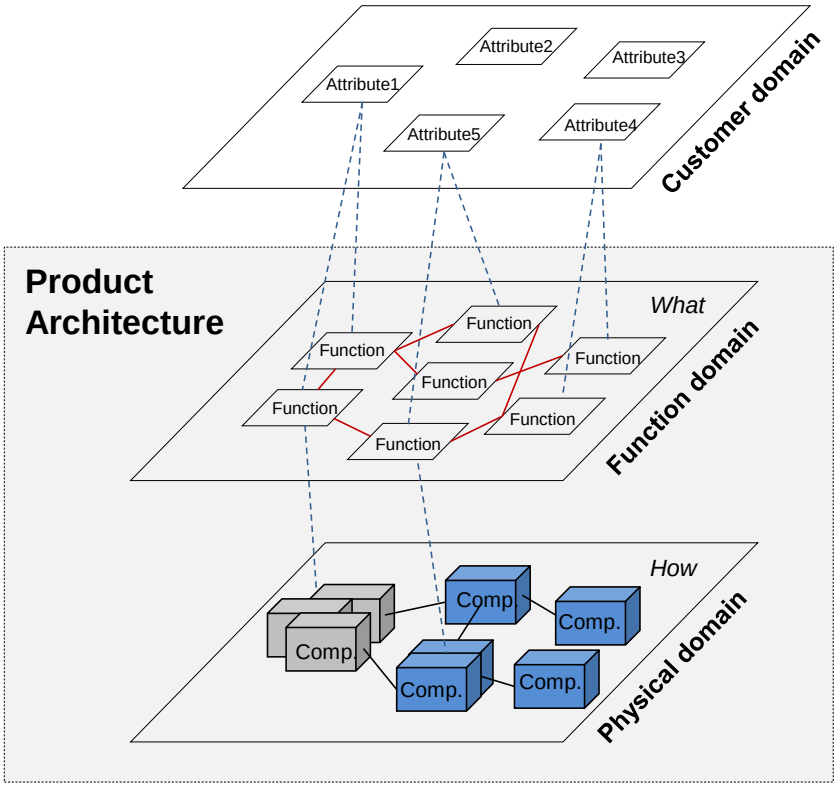


Figure 1-2. Product architecture penetrating design domains (Kang 2010)

each other through physical interface. Several physical components are grouped to compose a module, a separable unit that has one or more specified function elements. The grouping strategies of physical components should address the interdependencies and their characteristics. Therefore, product architecture organically streamlines design information on different domains and systemically amalgamate them. Consequently, the consistent and coherent design strategies throughout different design domains can be established based on the product architecture.

In incremental design, the architecture of the baseline system specifies design constraints for developing new system, as new system inherits core competent characteristics from the previous system. The architectural constraints come from the existing system can assure the reliability and feasibility for the new system. On the functional domain, the architecture of the baseline system is represented with the interrelationships among the functional elements (Maier and Rechtin 2000). The interrelationships are established by conducting repetitive experimentation and reflecting market response throughout product life cycle. Therefore, product planning based on the design space, which is defined by the interrelationships of the existing system, can reduce the risk of the entirely new product development. Consequently, the functional-element interrelationships of the existing product architecture define the design space the new system in incremental design.

On the physical domain, carried-over components from the existing system and their interdependency become the architectural constraints for the new system. In incremental design, the new components are added to the existing system in order to effectively satisfy requirement changes. Therefore, the new system should be carefully configured by addressing not only newly incorporated components and the carried-over components and their interdependencies. Especially, in incremental design for the modular system, which is composed with several modules, the carried-over module's composition and their interdependency can be architectural constraints, because organization of the firm and production system are aligned to the module units.

In incremental design, based on understanding design constraints that come from the product architecture of the existing system, design solutions for the new system should be explored with adopting new requirements from customers, regulators, and the business environment (e.g. manufacturing constraints). In detail, on the functional domain, a consistent set of design targets for new product should be determined on the foundation of the interrelationships between previous product's specifications. Meanwhile, on the physical domain, the arrangement of components for the new system, which add new components to the existing system in order to meet new requirements, should be established based on the carried-over components/modules and their interdependencies.

1.3. Design strategies on the existing architecture

This paper proposes architecture-based strategies on different design domains for incremental design. On the functional domain, a methodology is proposed to determine design targets on the early stage of development process with considering market environment and design constraints, which represent interrelationships between specifications (functional elements) of the existing products. On the physical domain, this paper proposes a methodology to rearrange (re-architect) an existing system that has modular architecture when new technologies are to be infused via a set of new components.

On the early stage of incremental design, determination of design targets based on the existing interrelations among the functional elements is a

primary concern. Especially, as product systems get more complex, the design targets should be clarified and consistent over the entire system. In incremental design, the feasible design space for designing new product is defined by the interrelationships between specifications of the existing products. The new design solutions that reflect changed requirements are determined within the design space. In the present paper, a methodology is proposed to identify interrelationships among functional elements by analyzing historically launched products in data-driven manners; the interrelationships represent the existing system's product architecture on the functional domain (Kang et al. 2013). Thereafter, the design targets, what to achieve in the new product, are determined to maximize the customer utility within the feasible design space bounded with the interrelationships.

When design targets are specified on the functional domain based on the existing system, they should be cascaded to the physical domain through geometrical embodiments. In incremental design, the manufacturing company critically assessed the previous product generation to understand which parts could be carried over, which would need modification and where innovation would be required. If the newly established design targets could not be satisfied with the existing components, new components should be incorporated by geometrically embodying new technologies. Iterations are required if the technology cannot meet the requirements or the results clash with the existing geometry.

On the physical domain, this paper proposes a methodology to re-architect an existing system that has modular architecture when new technologies are to be infused via a set of new components. The proposed method explicitly recognizes the existing system, as the foundation of the new system, focuses on the transformation of the existing architecture into the optimal architecture of the new system. Vast amount of prior research on designing modular systems or building product platforms have proposed numerous methodologies to determine the optimal architecture for developing new products, implicitly acknowledging the existence of the previous design. Although it is imperative to determine the goal of the new architecture, the paths of transformation, from the existing to the optimal architecture for a new system, can be exceedingly varied; furthermore, the optimal architecture itself should depend on the transformation path selected to meet the new requirements. Therefore, the method proposed in this paper models the transformation of an existing architecture by reflecting required changes. The present study determined the optimal architecture by consideration of the relevant transformation characteristics.

1.4. Structure of Thesis

A brief description on subsequent chapters is presented in this section. Chapter 2 provides review on previous literature related to design strategies on new product development. Chapter 3 proposes a new methodology to determine

design targets on the early stage of development process with considering design constraints and market environments through a vehicle planning case. Herein, the design constraints are established by statistically analyzing historically launched-product specifications. Chapter 4 proposes a methodology to re-architect modular systems in incremental design, which can be utilized on the physical domain. Meanwhile, as the system get more complex, the developed re-architecting methodology might confront the combinatorial explosion. To surmount this difficulty, Chapter 5 presents a genetic algorithm for the re-architecting methodology; thereby, the re-architecting methodology can be applied to complex systems in real world. Consequently, in Chapter 6, conclusions and contributions are summarized. Implications of present methodologies and more complicated issues that this research lacks will stimulate further research.

Chapter 2. Literature Review

2.1 Determination of design targets on the early stage of development process

Among the earliest conceptualizations of engineering-specification generation in product planning were those of Hauser and Clausing (1988) and Cohen (1995), who effectively framed the problem in terms of information-conversion processes requiring tools such as house of quality and quality function deployment (QFD). Recent research efforts have linked engineering optimization models with market demand models for the purpose of the design products and product lines. De Weck (2006) determined product platform extent by adopting an end-to-end modeling framework that connects six different domains: product architecture, performance engineering, value modeling, market demand modeling, manufacturing costing and investment finance. Gu et al. (2002) built separate models for marketing and engineering decisions and coordinated them for multidisciplinary design optimization. Michalek et al. (2011) proposed a model that links marketing and engineering product design decisions with analytical target cascading (ATC; which models the propagation of top-level targets to appropriate specifications in a consistent and efficient manner (Kim 2001)). Hoyle et al. (2010) developed an integrated model for multiple hierarchy of complex systems in order to create a link between

the qualitative attributes and the determined optimal value of a vehicle package's dimensions.

The integration of models from various disciplines into a product-design system, a large and complex undertaking, will incur significant difficult. First, it is nearly impossible to theoretically comprehend all of the relations among design parameters composing complex product systems. Although theoretical engineering relations can be understood, in designing parts of complex system, it would be limited to deductively design the entire system. The second significant difficulty is the fact that in cases of complex product design, direct translation of customer requirements to design parameters should be limited. Indeed, recent studies can be appropriate to the design either a limited number of parts of a complex system or of simple products; however, their practical application to new-product planning, specifically to the determination of product-level specifications requiring an understanding the entirety of a system's relations, might be nearly impossible.

Our research surmounts the obstacles in planning complex product by eliciting design information from large amounts of real data (vehicle-specification and market-sales data). In product development, recent efforts have attempted to capture essential information from large amounts of data. A large data set of change request information was examined to compose product architecture in complex system (Giffin et al. 2009, Gokpinar et al. 2010). A recent stream of marketing research on preference measurement has focused on identifying new sources of data, such as combining scanner-based data with survey data (Horsky et al. 2006), market share information (Gilbride et al.

2008), and online product reviews (Lee and Bradlow 2007, Archak et al. 2011). On this stream of research, this study (1) elicited interrelationships between specifications from real vehicle data to define design space and (2) determined how people value different attributes in a vehicle model with real market sales data. The interplays among elicited design information are captured and utilized to determine a consistent set of design targets in vehicle planning.

To summarize, the main contribution of this paper is to show how design information embedded in real data can be utilized in vehicle planning and to determine a consistent set of design targets by coordinating those design information on moderate level. Simultaneously, the present research aims to highlight the value of data-driven product planning to complex product developments, which include difficulties of understanding overall system on early stage, to secure both the objectivity and the consistency of design-target establishment.

2.2 Product architecture

A product can be thought of in both functional and physical terms. The functional elements of a product are the transformations that contribute to the overall performance of the product. The physical elements of a product are the parts, components, and subassemblies that realize functional elements in the physical domain. With these terms, the product architecture is defined as (1) the arrangement of the functional elements, (2) the mapping from functional elements to physical components, and (3) the specification of the interfaces

among interacting physical components (Ulrich 1995, Ulrich and Eppinger 2007).

The physical elements of a product are typically organized into several major physical building blocks, called modules. Each module is composed of a collection of physical components that implement the functional elements of the product. With admitting the concept of the module, the product architecture can be understood as the scheme by which the functional elements of the product are mapped or arranged into modules and by which the modules interact.

The most important characteristic of a product architecture is in its modularity (Baldwin and Clark 2000). The *modularity* can be defined as *the degree of grouping that puts tightly connected physical components in the same boundary with no dependencies between these boundaries*. In other words, the scheme of the boundaries that bind the frequently interacting components determines the modularity of a product. The scheme can be evaluated by referring two properties: (1) how many (much) interdependencies are (is) in same modules; (2) how many (much) interdependencies are (is) outside of the modules (MacCormack et al. 2006). The higher modular product should tightly interact within each modules but loosely interact between the modules.

According to how modules of a product implement functional elements, the product's architecture is classified with *modular architecture* and *integral architecture*. In modular architecture, modules implement one or a few functional elements in their entirety, and the interactions between modules are well defined and are generally fundamental to the primary functions of the

product. In an integral architecture, functional elements of the product are implemented using more than one modules, and the interactions between modules are ill defined.

How to compose modules with physical components and how much modularity to impose on the architecture are related to several important issues to the entire enterprise: product change, product variety, component standardization, product performance, manufacturability, and product development management (Ulrich and Eppinger 2007):

Product Change: when new required design changes are infused to the system, the changes are propagated through interdependencies (e.g. physical interfaces) to the entire system (Clarkson et al. 2004, Suh et al. 2010). In this situation, the modular architecture effectively isolates the propagations without necessarily affecting the design of other modules or components. Meanwhile, changing in integral architecture may influence many functional elements and require changes to several related modules.

Product Variety: Because a modular architecture has loosely interrelated modules, product build around modular architectures can be more easily varied by replacing or augmenting new modules to the existing systems without adding tremendous complexity to the manufacturing system (Baldwin and Clark 2000).

Component Standardization: If the various products are share common components or modules, then the modules or components can be standardized

and used in several different products (Meyer and Lehnerd 1997). This standardization can be also applied to physical interfaces among the components or modules. The standardized interfaces isolate changes occurred in a certain modules from propagating to related modules or components.

Manufacturability and Product development management: The product architecture also directly affects the ability of the team to design each module to be produced at low cost. Especially, in modular architecture, organization of the firm can be structured according to module units (Henderson and Clark 1990), and each group can independently design and develop a module with minimizing interacting other groups (Kim 2001). As the products get complex, the modular architecture can make product development management effective and efficient.

2.3 Architectural investigation in incremental design

The bulk of the engineering design literature has focused on product modularity in that modularity allows a designer to control the degree to which changes in processes or requirements and gives designers more flexibility to meet these changing processes (Gershenson et al. 2003). The benefits of modular system range from development to production. They include: (1) component economies of scale due to the use of components across product families and generations, (2) ease of product updating via adopting new technologies, (3) increased product variety with a small set of components, (4) decreased order

lead-time due to fewer components, (5) ease of design and test (Allen and Carlson-Skalak 1998, Martin and Ishii 2002), and (6) ease of service due to differential consumption (Ulrich 1994).

In the consideration of dynamic possibilities modular design offers, flexibility in design is the most important benefit (Baldwin and Clark 2000). Clearly, modularity allows for flexibility in function and flexibility in meeting end-user requirements. Over the life of a product, modular system can respond to changes in functional requirements over time due to various reasons like changes in customer requirements, regulations, and innovative technology introduced (Tate et al. 1998). Reconfiguration by changing the arrangement and adding new modules can realize the required functions on the foundation of the existing product (Sosale et al. 1997, Campagnolo and Camuffo 2010).

Whilst a number of paper have had a general consensus on the benefit and importance of reconfiguration in modular systems, there has been little effort to make algorithm or methodology for how to re-configure or re-arrange (Gershenson et al. 2004, Campagnolo and Camuffo 2010). The early modularity research focused on clarifying the definitions of related concepts and the benefits of modularity (Gershenson et al. 2003). Relatively recent studies made efforts on developing the method of product representation for modular design, measures of modularity. On the foundation of modularity concept, several studies proposed modular product design methods utilizing modularity measure, dependency and similarity (compatibility) on intra/inter module (Gershenson et al. 1999, Fixson 2006, Mikkola 2006). Nonetheless, the proposed methods give a single, final result do not take into account the design

changes that are necessary to accommodate each reconfiguration and to incorporate new components for changed requirements. Understand of induced changes on the process of rearrangement of the existing system is essential to design evolved product for new requirements. The present paper understands mechanisms of induced changes occurred in reconfiguration and finds alternative architectures for changed requirements with considering the induced changes and the modularity of the entire system. This study contributes to the modular design methodology in that it systemically and explicitly takes into account the design changes to establish design-evolution strategy of the modular systems.

Among researchers, awareness of the importance of modification or redesign of existing systems to meet new requirements and restrictions has been growing. Frameworks for determining optimal design parameters in the feasible design space in the early, conceptual design phase have been proposed. Nunez et al. (2012) proposed a method that determines the design parameters by exploring the feasible design space based on an isoperformance methodology (De Weck and Jones 2006). Meanwhile, Mavris and Kirby (1999) proposed the technology identification, evaluation, and selection (TIES) method to identify technically feasible and economically viable design concepts based on their baseline system, commercial airplanes. They devised a "technology impact matrix" to predict the impact of the addition of new technologies on the design parameters of the baseline system. Otto and Wood (1998) introduced a new reverse engineering and redesign methodology for continual pre-existing-product evolution. Through reverse engineering, customer needs are

formulated and a functional model is created; the gap between the new requirements and the existing system is defined and then filled with newly designed solutions.

Common to all these research streams and others relating to conceptual design can have limitation the solutions are handled on functional domain; therefore, how to deal with physical components constituting the existing system remains unanswered. In a bid for solving, Smaling and De Weck (2007) approached the incremental design problem from the standpoint of the assessment of physical-component-layer risks and opportunities. Focusing on the incorporation of new-technology into existing systems, they employed a component-based change design structure matrix (simply, "delta DSM") to quantify the number of changes required to incorporate each technology concept into the baseline system in the physical domain. On the basis of the assessment, they applied the method to a hydrogen-enhanced combustion engine. Similarly, Suh et al. (2010) quantified and assessed the impact of technology incorporation early in the product-planning cycle, and applied their results to an actual complex printing system. Despite the emerging comprehensive assessment/diagnosis methodologies of required changes however, how to actively deal with those changes based on an existing system has not yet been discussed.

According to all of these research streams, this paper proposes a methodology that redesigns an existing system in order to accommodate required changes. The present research focused specifically on re-architecting a product composed of pre-defined modules. To that end, the design structure matrix

(DSM) technique was adopted to characterize and analyse the product architecture. Steward (1981) and Eppinger et al. (1994) used a DSM to highlight the inherent architecture of a design, specifically by examining the dependency existing between components. MacCormack et al. (2006) utilized a DSM technique to devised two indices, propagation and clustered cost, suitable for comparison of the modular architectures of two software products; the Linux operating system and the Mozilla web browser. Additionally, in order to systemically generate alternatives for the new system, this study defines re-architecting operators with DSM based on prior researches. Baldwin and Clark (2000) defined six modular operators to describe all possible evolutionary paths for modular design. Gamba and Fusari (2009) proposed a methodology based on the modular operator concept for quantification of the value contribution of the modularization process and of individual modules. These approaches focused on describing and analysing state transitions of modularity from the macroscopic perspective of the entire system. The present study contrastingly focused on possible changes of product architecture that can be defined according to the dependencies among component elements that constitute modules, when new technologies are to be infused via a set of new components.

Chapter 3. Data-driven Optimized Vehicle-level Engineering Specification

In an increasingly globalized market environment, designing automobiles for key target markets is a daunting challenge. What makes vehicle planning even more problematic are various factors such as engineering feasibility, market-environment changes, regulatory enforcement, and the ongoing advance of technology. In this context, the complexity of automobile design arises from the fact that both target-market customer preferences and engineering design constraints need to be taken into account in a quantitative and balanced manner. This paper introduces a balanced model of vehicle-level specification determination based on maximization of automobile marketability within the given design constraints. The proposed model takes into account the interrelation of specifications, technology advance, vehicle style concepts, and customer preference changes as influenced by the fluctuation of market environments. The model was validated with reference to two mid-size sedans currently sold in the U.S. market.

Keywords: Vehicle-level specifications; vehicle planning; automobile design; design constraints; customer preference; optimization

3.1. Introduction

As the auto industry becomes ever-more globalized, automakers, if they are to successfully gamer a large share of the world market's proverbial pie, must take multifarious factors into consideration in planning for new-car models. Each global carmaker has its own identity in terms of design, aesthetic style, and technological prowess, among other measures. The market environment, as manifested for example in the price of oil, also influences automobile design. Indeed, as oil prices skyrocketed in mid-2008, consumer preferences shifted toward compact vehicles (Haugh et al. 2010). Also, as new legislated global standards impact not only on the issue of fuel economy but also on sustainability as well as environmental-costs control within the manufacturing and distribution realms (Lee and Cheong 2011), vehicle innovation will be driven by the legislation (Newbury and Lewin 2008). Meanwhile, automobile technology continues its advance. Powertrain related innovation, for instance, has improved engine performance and efficiency. Moreover, alternative propulsion systems also have received considerable attention.

The vehicle development process is particularly challenging in that it involves thousands of engineers spending years designing, testing, and integrating hundreds of thousands of parts (Gokpinar et al. 2010). Being different to other complex system development, in automobile industry well-targeted development is becoming more important according to an increasing fragmentation of the automotive market; indeed, the number of individual models for sale in the U.S. rises each year from 33 in 1947, to 198 in 1990 to an estimated 277 in 2009 (De Weck 2006). In these circumstances, how to systemically

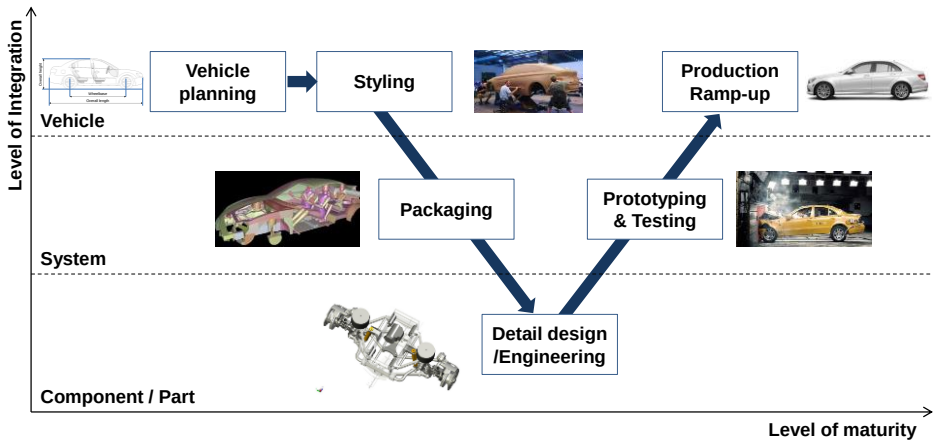


Figure 3-1. Vehicle development process

conduct a series of the vehicle development processes is a key for market success. Figure 3-1 shows the typical vehicle development processes (Weber 2009). Among them, vehicle planning is especially imperative in that it triggers the entire series of processes with all the associated costs and risks; particularly, a consistent set of design targets is determined in the planning phase. In nature, the initial vehicle planning phase should fulfill the role of an interface between marketing issues and engineering concerns as translating customer requirements from the language of marketing to the language of engineering. In order to enable development of a systematic vehicle planning methodology, not only should these two sides by scientifically analyzed on their respective merits but also, information should be organically exchangeable between the on the moderate aggregation level. Consequently, a key challenge in new vehicle planning is coordinating different design information come from various disciplines and pursuing a consistent set of design goals.

This involves two fundamental problems: (1) how to characterize primary design concerns with key factors, and (2) how to coordinate the interplay among the key factors.

In practical cases, given the dauntingly multifarious factors to consider, new-vehicle planning has been conducted, based on the intuition of experts or fragmentary benchmarking studies of competitors' vehicles, in an empirical and qualitative manner. However, the results of such approach vary according to the analytical capabilities of the individuals involved. Moreover, trend analysis of several generations of competitors' vehicles has limitations with respect to comprisable items and the precision of results. To determine a consistent set of quantitative targets that all members can agree on, a systematic and scientific methodology in vehicle planning should be established that can utilize moderate levels of data.

This paper developed a novel, applicable method for essentializing design information from markets and engineering concerns in vehicle planning and analyzing them together on a common board to determine a consistent set of design targets. In marketing issues, the values that customers assign to vehicle attributes are quantified by statistical techniques of a demand analysis of market share data. In engineering domain, the feasible design space, which is represented as relationships among vehicle-level specifications, is defined by utilizing real specification data of the launched vehicles. Consequently, a consistent set of vehicle-level specifications is determined by maximizing the customer-utility within the defined feasible design space.

3.2. Research overview

3.2.1 Data set

Motor company A originally posed the question, How can the vehicle-level specifications be determined so as to establish consistent design targets in the early stages of the vehicle development processes? And, relatedly, how can close competitors' vehicle specifications be anticipated so as to make more informed strategic decisions and, thus, enhance the competitiveness of our next-generation models? For instance, Toyota, one of the major competitive automakers in the U.S. mid-size sedan market, launched Camry 2007 with improved fuel economy, decreased overall height, an extended wheelbase, and widened overall width, based on Camry 2002. These changes were based on two main pillars: changing customer priorities relating to fluctuating market conditions such as the price of oil, and consistent advances in the relevant powertrain and engine technologies. Hence, to predict a next- generation vehicle, designers should take into consideration not only market changes but also the design feasibility in balanced manner.

This paper proposes a comprehensive methodology for explicitly determining a consistent set of design goals in the early stages of the design process. The methodology utilizes real specification data of launched vehicles in U.S. market and their market sales with using statistical techniques to extract essential design information. Significantly, the methodology effectively captures the interplay among key factors in preliminary vehicle design: design feasibility constraints, market demands, and economic conditions. Rather than

attempt to model a complex system with limited information that contains potentially large uncertainties, the proposed approach avoids the difficulties of understanding an entire complex system by determining only the moderate level of specifications using statistical analysis of historical data. Moreover, its quantification of revealed customer preferences as expressed in sales data resolves the confusions in vehicle planning arising from the translation of customer requirements to engineering specifications.

For the vehicle-planning methodology, this paper built two kinds of

Table 3-1. Abbreviations

OAL	=	overall length
OAW	=	overall width
OAH	=	overall height
Wbase	=	wheel base
HP	=	maximum horsepower
CFE	=	combined fuel economy
Displ	=	Displacement
Tq	=	maximum torque
CWT	=	curb weight
RPM	=	RPM for maximum horsepower
HeadFRT	=	front headroom
HeadRR	=	rear headroom
LegFRT	=	front legroom
LegRR	=	rear legroom
WTFRT	=	front wheel track
WTRR	=	rear wheel track
FDR	=	final drive ratio
TM	=	transmission
TireWid	=	tire width

panel data. First, the author collected data on 19 vehicle-level specifications, presented in Table 3-1 with their abbreviations, based on 150 the mid-size sedan models on the U.S. market from 1990 to 2009. This panel data was utilized to extract vehicle-specification interrelationship in various engineering design perspectives. Second, to understand customer preferences on vehicle attributes in target market, the author built another panel data constituting observations, car models, sales data and multiple vehicle-level specifications over multiple time periods for the mid-size sedan market. Specifically, 23 vehicle-level specifications, which incorporates models' brand information to the 19 vehicle-level specifications, were collected for different models and months for the years 2003 to 2009. The author regarded an automobile sold in a different month as a different model, even if the specifications were exactly the same. Moreover, if a certain model had several trims, each trim was regarded as distinct. Accordingly, the panel data included 770 observations. Various data sources was referred to compose these two panel data: 'Hyundai motor company's database', 'Automotive news data center', 'autos.msn.com', and catalogs of mid-size sedans in the U.S. market.

3.2.2 Design information and their interplay

Determination of vehicle-level specifications involves relevant design information in two domains (Suh 2001): the customer domain and the engineering domain. The customer domain is characterized by the key attributes customers look for in a new automobile and their preferences respecting each

attribute. The engineering domain is characterized by specifications and constraints. Specifications are sets of design targets to be achieved in new vehicles; constraints confine those specifications to the feasible design space. Attributes and specifications, being in different domains, are interlinked via a design matrix that defines their relations. Through the design matrix, information on the customer domain flows into the engineering domain, and vice versa. To obtain information on the customer domain, the author used mapping equations to define relations between a set of attributes and their relevant specifications.

The present research identified five key attributes based on expert opinions. The author conducted a series of interviews with chief engineers in charge of planning for mid-sized sedans and was able to reach at the consensus of those five key attributes: accelerating ability, combined fuel economy, width distribution, roominess, and sporty look. The two major concerns of the chief engineers were dynamic performance and exterior design. While the ac-

Table 3-2. Mapping functions between specifications and attributes

i	**Attribute,** A_i	**Function of specifications,** $f_i(X)$
1	Accelerating ability	$\frac{CWT}{HP}$
2	Combined fuel economy	CFE
3	Width distribution	$\frac{OAW}{Wbase}$
4	Roominess	$OAW \times Wbase$
5	Sporty look	$\frac{Wbase}{OAL}$

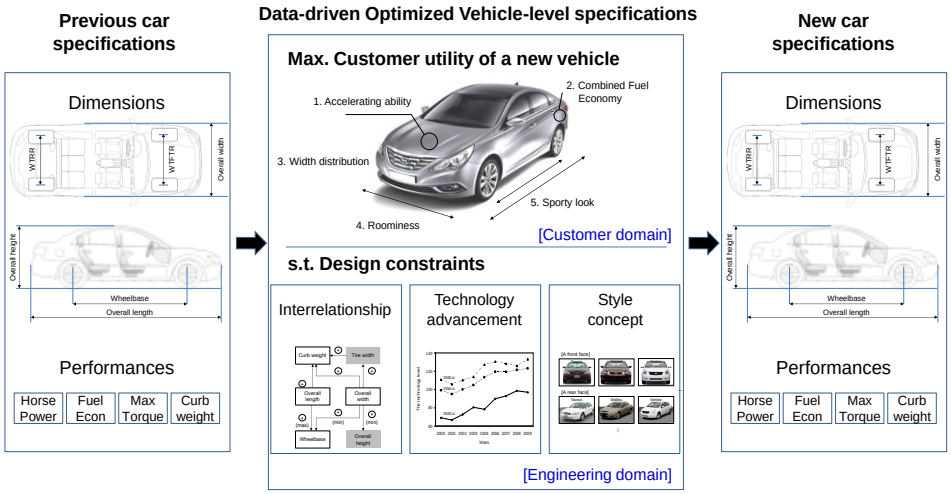


Figure 3-2. Data-driven optimized vehicle-level engineering specifications (DOVES)

celerating ability and combined fuel economy are key factors for dynamic performance, the other attributes are for exterior design. Those key attributes (*A*) were then expressed in terms of the vehicle-level specifications (*X*), with the help of expert designers, as shown in Table 3-2.

Figure 3-2 overviews a new methodology based on the understanding of two domains, namely data-driven optimized vehicle-level engineering specifications (DOVES), which identifies design targets in a consistent manner. Briefly, DOVES uses statistical analysis of historically launched vehicles to define design space according to engineering domain specifications; that is, it does not attempt to understand all of the mechanical relations among those vehicle specifications, but rather it assumes that the designs of historically launched vehicles have been mechanically feasible and sound. The design constraints are constructed with respect to (1) the interrelationships between specifications, (2) the advancement of technology, and (3) the style concept.

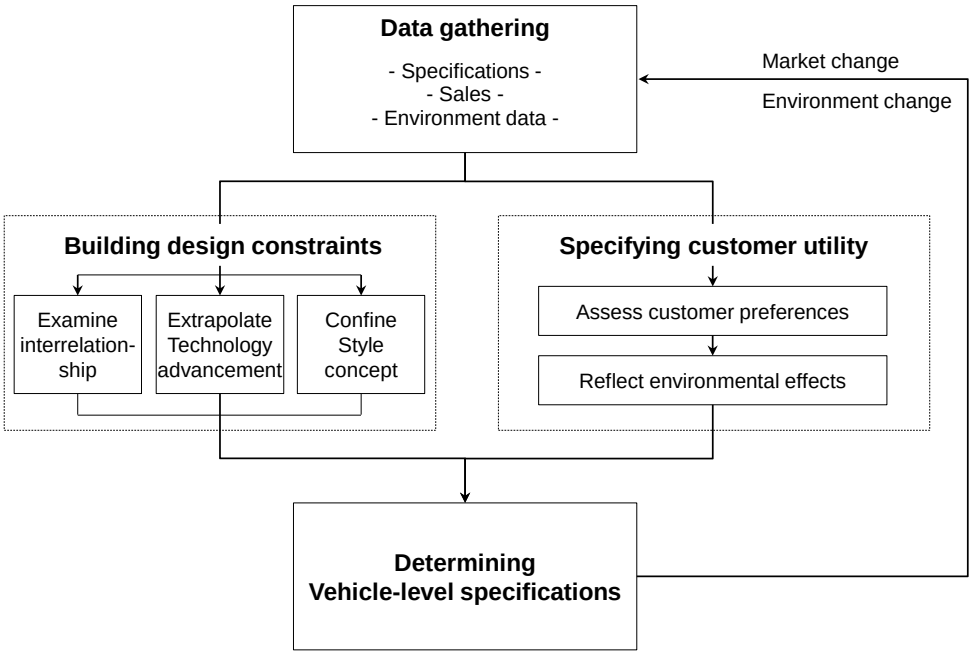


Figure 3-3. Overall research processes

The customer utility derived from a planning vehicle is determined by measuring how people value different attributes. Within the defined design space, DOVES determines a set of specifications that maximize the customer utility with mapping functions between specifications and attributes.

Overall research processes to build DOVES methodology is represented in Figure 3-2. Data is firstly gathered to compose two kinds of panel data. In addition, data of environmental factors, which influence customer behaviour in automobile market, is also collected. Based on the collected data, essential design information on both engineering and customer domains are extracted in parallel. In building design constraints on engineering domain, interrelationships among vehicle-level specifications define the feasible and

strategic design space. Meanwhile, customer utility derived from a new vehicle is specified by customer preference assessment with reflecting environmental effects. Extracted key factors are put into on a common board, and a consistent set of design targets is determined by optimization technique. A series of procedures can be repeated, when there is changes of competing automobile models in target market or changes of market environments, through updating and reconstituting data base. In the following sections, the establishment of the design constraint model and value function of the customer utility of a new-vehicle concept are discussed in detail.

3.3. Proposed approach: models and procedures

3.3.1 Building design constraints

3.3.1.1 Specification bounds and interrelationship

The author began by conducting a correlation analysis of all specifications to gain a summary understanding of the relations among the vehicle-level systems. Based on these statistical correlations, the author divided the collected vehicle-level specifications into two groups by dimensions and performance. Then, the author endeavoured to find the statistical relations among the specifications in each group. Based on interviews of body-, interior-, chassis- and package-design engineers, the author established a reducible schematic diagram indicating the directions of the influences among the specifications. A

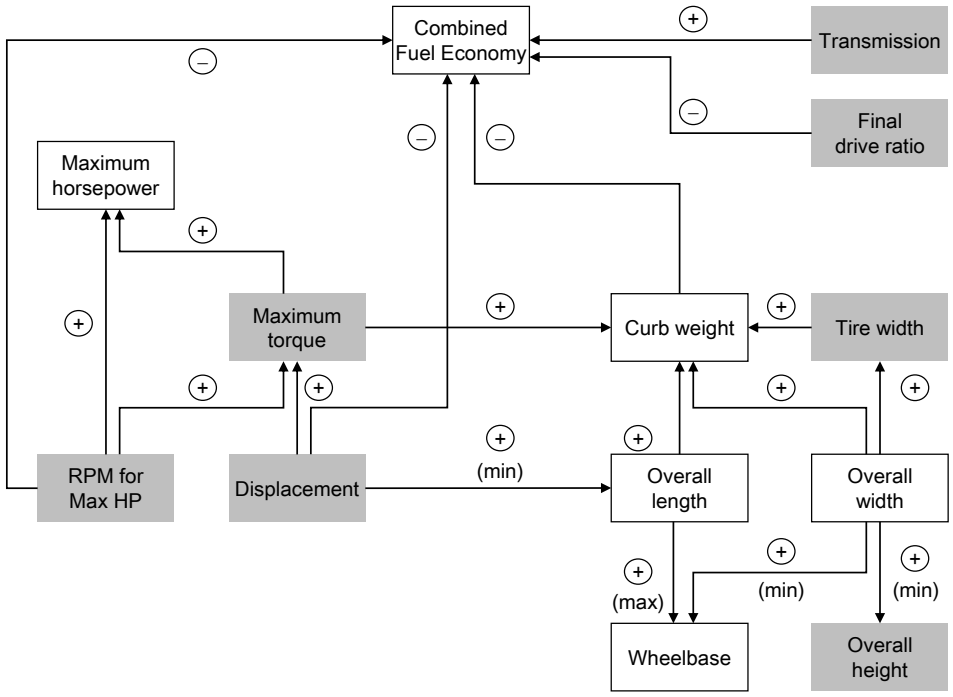


Figure 3-4. Interrelationships among vehicle-level specifications

statistical analysis of the characteristic variables for verification of those influences was conducted, followed by an omnidirectional statistical analysis of all of the specifications (see the Appendix A).

Specifically, a regression analysis was conducted to find the statistical relationships among the vehicle-level specifications indicated in the reducible schematic diagram. To find the best set explaining each of three performance specifications, which are horsepower (HP), curb weight (CWT), and combined fuel economy (CFE), one or more specifications as independent variables are iteratively analysed in various combinations. In the result, three such relationships were found, for an overall analytical performance above 90% of adjusted

R^2. Second, the author drew a scatter plot to display the values of the two vehicle-level specifications for a set of data. Based on a scatter plot of historical vehicle models, correlations between the two dimensional specifications, according to whether the dot pattern sloped from the lower left to the upper right or oppositely, were found. Particularly, the limiting value that one-dimensional specification can have with respect to another dimension is found and represented as an inequality equation in the form of line bounding the dots in a scatter plot. Figure A2 in the Appendix shows scatter plots of various combinations of dimensional specifications and the lines bounding their dots.

Figure 3-4 shows the interrelationships among the specifications in the form of rectangular surrogate boxes, with arrow lines representing the directions of those relations. The interrelationships among the specifications were bounded by two types of constraints: inequality and relational equality equations. In Fig. 4, the positive sign (+) indicates that if one characteristic variable is increased, the effected variable also is increased. For the reverse case, the minus sign (-) is marked. "(max)" and "(min)" represent inequality relationships ("≤", "≥") between two specifications. Displacement indicates the overall length of an automobile, a lower bound with a minimum inequality; overall width provides the minimum constraints of the wheelbase and overall height; overall length bounds the maximum constraint of the wheelbase. These inequality constraints could be obtained by plotting the historical data and deriving the bounds from those plots. Thereby, the author could understand which among the vehicle-level specifications are related to each performance specification: Combined fuel economy, maximum horsepower, curb weight, and

maximum torque. Thereafter, through regression analyses with the best subsets, we found statistically significant relations for the performance specifications. Those interrelations build up the foundation of the system architecture with inequality relations, as shown in Fig. 3-4. The above-noted constraints were converted to inequality and equality equations as shown in Table 3-3.

3.3.1.2 Technology advancement

To predict the progress of common-use technology, the author introduced a new concept, advanced technology's path for the fuel economy and horsepower that represents the tendency of an engine technology's evolution, and used it as a reference according to the assumption that common-use technology should follow advanced technology. By superimposing the point of common-use on the advanced technology path, a technology gap can be inferred. According to this technology gap, the new vehicle concept's technology level

Table 3-3. Linear relationships among vehicle-level specifications

	Specifications (x)																										
	MY	OAL	OAW	OAH	Wbase	Head FRT	Head RR	Leg FRT	Leg RR	WT FRT	WT RR	Displ	HP	Fuel Eco	Tq	RPM HP	CWT	FDR	TM	Tire Wid	Toyo -ta	EU Dm	Honda Dm	Nissan Dm	GM Dm	WD	Tq *RPM	
	Interrelations (Aeqvar)																										beqvar
Equality equations (Aeqvar*x = beqvar)	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	= x_HP0 *HPInflation
	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	6.43	2.46	0	0	0.00	= 7.19
	0	0.17	0.83	0	0	0	0	0	0	0	0	0	0	0	11.11	0	-1	0	0	1.97	0	95.82	0	-35.09	0	0	0	= 1530.2
	0	0	0	0	0	0	0	0	0	0	0	0	0	-1/FEInflation	0	0.00	0	-0.31	0.51	0	1.10	0	1.20	0	0	41.57	0	= -28.519

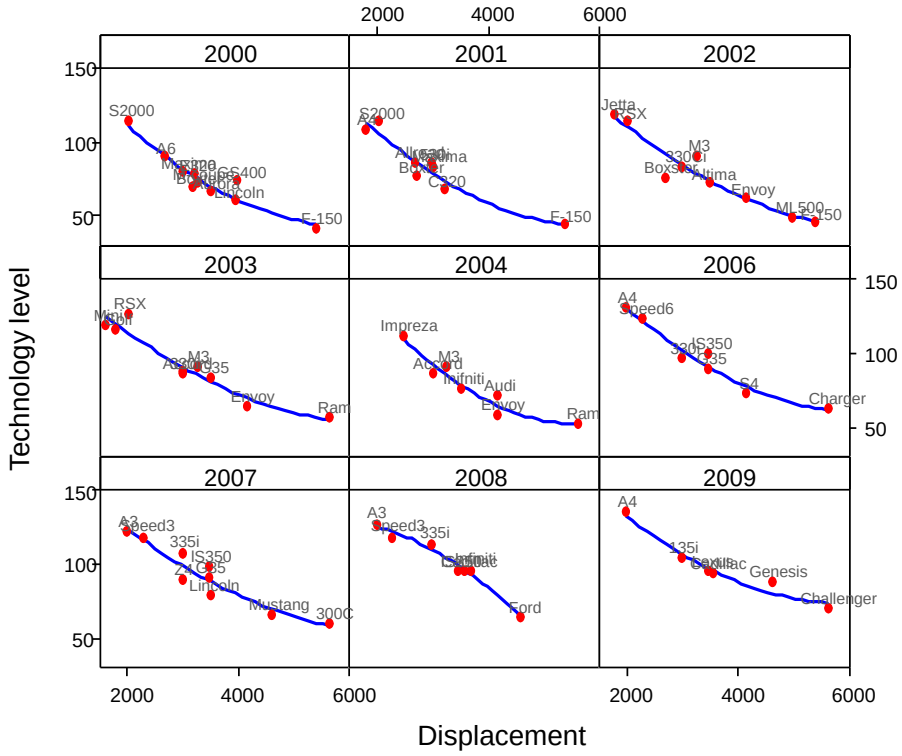


Figure 3-5. Scatter plots of automobiles adopting Ward's 10 best engines

is identified as a point on the advanced technology path.

To identify the new index for the technology advancement of dynamic performance, it is required to find an appropriate combination of performance specifications that linearly increases as time goes on. The dynamic ability of vehicle is determined by performance-related specifications, such as maximum horse power, curb weight, displacement, and combined fuel economy.

Advancement tendency of each performance specification is unpredictable because several factors, such as environmental situation, brand image, and market trends, influence on the tendency. Meanwhile, those specifications are closely related to each other. Specifically, based on pairwise regression analysis of performance specifications, it was found that displacement affects to maximum horse power and combined fuel economy. In addition, curb weight intimately influences on combined fuel economy. Even though it was hard to identify individual advancement tendency, the advancement of the relationship among the performance specifications could be elicited. The present research grasped the relationship with defining a new index, which is composed of horse power, combined fuel economy, curb weight, and displacement. The relationship linearly advanced in time as shown in Figure 3-6.

To determine the advanced technology's progress, the author identifies the new index by analyzing historical specification data on dynamic performance. A new vehicle's advanced technology is defined as its adaptation of state-of-the-art engine technologies that outperform common-use technology in both fuel economy and horsepower. In fact, by analyzing historical data in the form of scatter plots of various combinations of performance specifications, the author could practically induce an index indicating FE-HP technology levels. First, we inductively found a trade-off between combined fuel economy and maximum horsepower. Second, the author found that displacement is proportionate to the value of combined fuel economy multiplied by the maximum horsepower. Finally, the author found that the square of the curb weight is in inverse proportion to $CFE \times HP / Displ$ (Appendix). Thereby,

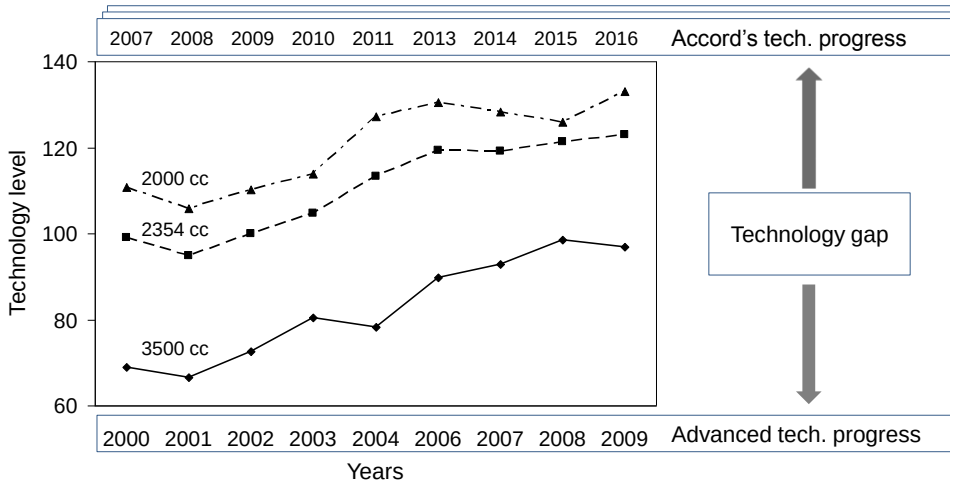


Figure 3-6. Progress paths of advanced technology

the author synthesized the above practical findings to a new index, the FE-HP technology level (L), defined as:

$$L = \frac{CFE \times HP \times \sqrt{CWT}}{Displ} \quad . \tag{3-1}$$

Note that Equation (1) is based entirely on statistically analyzing the value of the specifications themselves rather than understanding the mechanical characteristics of vehicle-level specifications of dynamic performance. Nonetheless, Equation (1) makes two key assumptions: if a car model achieves a certain level of fuel economy and horsepower with a lower displacement than others, it is regarded as representing a higher technology level; similarly, if an automobile model with a high curb weight achieves the same level of combined fuel economy as others, it is considered to occupy a higher technology level.

Based on the new index, historical auto models adopting 'Ward's 10 best Engines' can aid the identification of the progress of advanced technology. 'Ward's 10 best Engines,' Ward's AutoWorld magazine's annual list of the ten best automobile engines available in the U.S. market, has been compiled and published every year since 1994 (Wikipedia 2013). Figure 3-5 plots the FE-HP technology level (L) of all of the automobiles that had adopted Ward's 10 best engines. To identify progress path of the advanced technology according to a certain displacement, polynomial regression analysis relating the technology level to the displacement is conducted. As the statistical relation for each year is estimated, the advanced technology's progress path the given span of time can be established at the certain displacement. Figure 3-6 shows the progress path of the automobile models applying Ward's 10 best engines at the displacement levels 2000cc, 2354cc and 3500cc.

The technology gap is defined as the time lag between the continued use of common-use technology and the adoption of advanced technology. The technology gap for a certain vehicle can be deduced by finding the point on the technology progress path where the model's technology level corresponds to the advanced technology. For instance, the 2009 Accord model, with 193 PS, a combined fuel economy of 31.9 mpg, and a curb weight of 1489 kg for a displacement of 2354 cc, was computed to be on a technology level of 100.92. This corresponds to 2002's advanced technology level, which was identified as L=105.80 on the technology progress path for the 2354 cc displacement; thus the technology gap between the 2009 Accord and the advanced technol-

ogy was considered to be about 7 years. Although the technology gap can dramatically change over time, such as in the case of Hyundai Motors (see Section 3-5, "Model validation," below), the author assumed a technology gap that remains constant over time.

The technology gap establishes the target of a new model by selecting the corresponding point on the technology progress path. For the purpose of predicting a new 2013 Accord based on 2354 cc, the 2006 point (L=120) on the technology path can be used as the target technology level, because the Accord model's advanced technology lag is about 7 years. In this way, the target technology level becomes a design constraint. Previously, because the technology level index is established with combined fuel economy, maximum horsepower, curb weight and displacement variable, therefore, if the target technology level is set, those variables should be constrained to that. Note that the 'HpInflation' and 'FEInflation' variables were devised to reflect the requirements of increasing technology advancement. When the target technology level is set for the next generation model, the required technology advancement on both horsepower and combined fuel economy is melt into equality constraints through 'HPInflation' and 'FEInflation' shown as Table 3-3.

3.3.1.3 Style concepts

"Style" represents the strategic style concept trends associated with automakers, classes and times. The present research considered how exterior style ex-

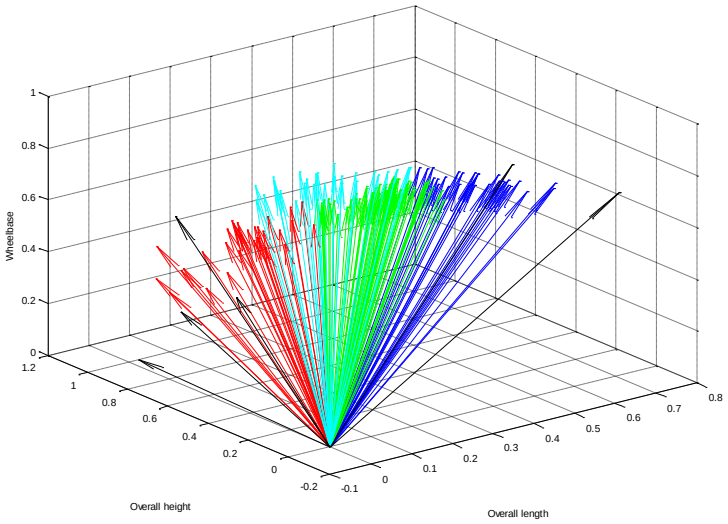


Figure 3-7. Dimension vectors for a side-face feature

pectations can constrain vehicle dimensions. The author assumes that an automobile's style has five features: front face, side face, rear face, roof line and front and rear room distribution. These are characterized according to the ratios of the relevant dimensional specifications. According to these ratios, the applicable historical automobile models are clustered so as to group the characteristic of each feature; thereby, several groups are distinguished, as shown in Fig. 3-7. Now, as designers selecting a style group for each feature, the dimension ratio characterizing the selected group becomes a constraint bounding the relevant dimensional specifications.

Five style feature viewpoints are defined in order to describe style concepts, as shown in Fig. 3-8. Moreover, the relevant dimensional features for each style feature is presented in Table 3-4. As each vehicle pursues a different

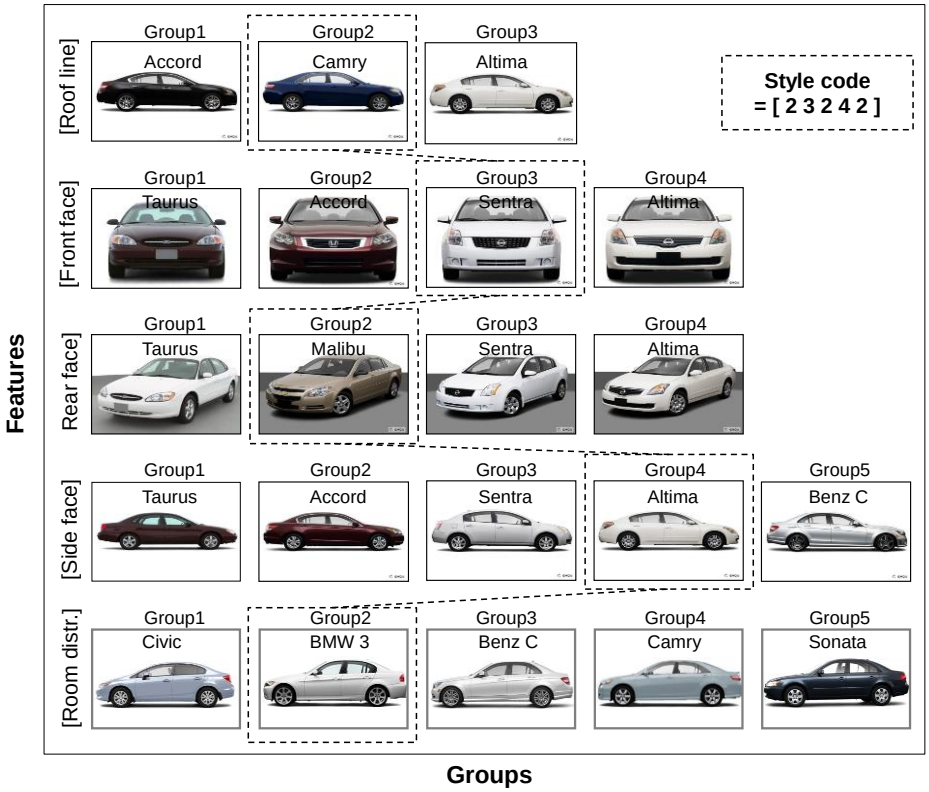


Figure 3-8. Several groups of each style feature and generation of style code

style concept sp ecified by dimensions, each style feature can be clustered into several groups according to the ratios of the relevant dimensional specifications. Ultimately, all automobiles should have ‘style codes’ designating which cluster of each feature they are in. Style code is defined as a set of group numbers representing the style concepts reflected in the features for a new vehicle model.

To group feature characteristics into clusters, the dimensional ratio for each feature is expressed as three-dimensional vectors based on historical data.

These vectors are clustered with a K-means clustering algorithm that defines a prototype of a centroid, which is usually the group mean, and is applied to objects in a continuous n-dimensional space (Tan et al. (2006)). Table 3-4 lists the centroid vectors for five style features and indicates the style concept corresponding to each cluster.

The clusters can be used as strategic constraints on the design of a new package layout, specifically by shaping conical field based on their centroids. Style constraints, according to a style code defined as the first step in planning

Table 3-4. Centroids and description for each style feature

A side face style	**Centroid vector**			
Cluster	**OAL**	**OAH**	**Wbase**	**Style concept**
1	1	0.722	1.048	Short wheelbase and low overall height
2	1	4.049	2.104	Extremely high overall height and long wheelbase
3	1	1.272	1.281	Ordinary overall height and wheelbase
4	1	1.687	1.335	Ordinary overall height and wheelbase
5	1	2.184	2.506	High overall height and wheelbase
A front face style	**Centroid vector**			
Cluster	**OAW**	**OAH**	**WTFRT**	**Style concept**
1	1	1.939	1.471	A bit high overall height and a little wide front wheeltread
2	1	0.818	1.037	Low overall height and narrow front wheeltread
3	1	1.298	1.221	A bit low overallheight and a little narrow fron wheeltread
4	1	3.454	1.959	Extremely high overall height and wide front wheeltread
Roofline style	**Centroid vector**			
Cluster	**OAH**	**HeadFTR**	**HEADRR**	**Style concept**
1	1	1.126	1.151	High front headroom and high rear headroom
2	1	0.674	0.951	Extremely high overall height and long wheelbase
3	1	1.016	0.624	High overall height and wheelbase
Room distribution	**Centroid vector**			
Cluster	**Wbase**	**LegFRT**	**LegRR**	**Style concept**
1	1	2.357	0.923	Spacious front legroom
2	1	0.879	0.786	A little small front legroom and rear legroom
3	1	1.289	0.588	Ordinary front legroom and extremely small rear legroom
4	1	1.019	1.127	Ordinary front legroom and rear legroom
5	1	1.638	1.002	A bit spacious front legroom and ordinary rear legroom
A rear face style	**Centroid vector**			
Cluster	**OAL**	**OAW**	**WTRR**	**Style concept**
1	1	0.945	0.869	A little overall width and narrow rear wheeltread
2	1	1.708	2.451	Wide overall width and rear wheeltread
3	1	1.088	1.284	Ordinary overall width and rear wheeltread
4	1	0.852	0.964	Extremely narrow overall width and narrow rear wheeltread

the style features of next-generation car models, are generated by shaping the conical space around the centroid vectors. The conical space is generated with reference to both a centroid vector and its maximum angle. The maximum angle is defined as the angle between the centroid vector and the dimension vectors that identifies the homogeneity of the dimension vectors with respect to the centroid vector. By defining the conical space that determines whether a certain dimension vector is in the same cluster or not, inequality constraints concerning the dimension vectors are obtained. Table 3-5 shows how the style code translates and operates as a constraint. Matrix *s* consists of centroid vectors according to the new style code, and *x* is a design vector of dimensional specifications. If the maximum angle is set to θ, the conical-shape-constrained space is defined as $x \cdot s \geq \cos\theta \cdot |x||s|$.

3.3.2 Maximizing customer utility based on customer preferences

The maximization of consumer utility subject to engineering design constraints is the purpose of determining vehicle-level specifications in the vehicle planning phase. The primary role of the value model is to translate the vehicle-level specifications into a scalar quantity called "customer utility".

Table 3-5. Style concept constraints

Style concept constraints $x \cdot s \geq \cos\theta\|x\|\|s\|$	**Specifications (x)**														
	MY	OAL	OAW	OAH	Wbase	HeadFRT	HeadRR	LegFRT	LegRR	WTFRT	WTRR	Displ	HP	FuelEco	
	Style centroid vector (s)														**cos θ|x||s|**
A side face	0	1	0	1.2722	1.2806	0	0	0	0	0	0	0	0	0	≥ 0.995|x||s|
A fron face	0	0	1	1.2988	0	0	0	0	0	1.2213	0	0	0	0	≥ 0.995|x||s|
Roofline	0	0	0	1	0	1.1257	1.1507	0	0	0	0	0	0	0	≥ 0.995|x||s|
Room distribution	0	0	0	0	1	0	0	0.8799	0.7856	0	0	0	0	0	≥ 0.995|x||s|
A rear face	0	1	1.0881	0	0	0	0	0	0	0	1.2839	0	0	0	≥ 0.995|x||s|

The customer utility is defined as the level of value that customers derive from attributes of a vehicle concept and is quantified as a function of both the value of those collective attributes and the customer preferences for each. The customer utility function of a vehicle can be represented as $\beta \cdot A$, where A is the vector of the attributes recognized by the customer, and β is the vector of the customer's preferences on them

3.3.2.1 Customer preferences assessment

This section discusses the estimation of the customer utility function in the U.S. mid-size sedan market. In the context of choosing products, such as automobiles, having differentiated attributes, customer utility estimation can be conducted with the discrete choice model. A vast area of market research on conjoint analysis (Green and Srinivasan 1978) and preference measurement has attempted to determine how people value different features in a product or service with using statistical techniques (Train 2003). Notably, Berry et al. (1995) developed the BLP model (named after authors) to empirically analyze demand and supply in differentiated product markets and then applied it to the U.S. automobile industry. In the present research, for simplicity, the author transformed the BLP model to a Logit model under the following model settings. The customer preferences on the five attributes are estimated with utilizing the market-demand panel data.

The level of utility that a consumer derives from a given automobile model is a function of both a vector of individual characteristics ξ and a vector of product attributes. The indirect utility function of consumer *i* from model *j*, can be represented as

$$U_{ij}(A_j, \xi_j, p_j; \theta) \equiv A_j\beta - \alpha p_j + \xi_j + \varepsilon_{ij}, \quad (3\text{-}2)$$

Where $\delta_j = A_j\beta - \alpha p_j + \xi_j$, the p_j is the price of product *j*, A_j and ξ_j are the observed and unobserved attributes of product *j*, α is the consumer's marginal utility from income, β is a vector of individual-specific consumer taste coefficients, and ε_{ij} is a mean-zero stochastic term. The utility function of customers is estimated with the logit choice probability function (McFadden 1974) based on ordinary least squares. Based on the market-demand panel data, the author could convert the specifications to attribute values using the mapping functions shown in Table 3-2. Most notably, roominess, sporty look and width distribution were converted to normalized values with the mean and standard deviations of competing models. To estimate customer utility function, there should be no multicollinearity among the five key attributes, which are predictor variables of customer utility in a multiple regression model. Multicollinearity refers to a situation in which two or more explanatory variables in a

multiple regression model are highly linearly related and correlated. Multicollinearity affects calculations regarding individual attribute factors, so it is hard to understand how designed value of an individual vehicle attribute contributes to customer's satisfaction. To detect multicollinearity, tolerance, which is obtained from the coefficient of determination of a regression of explanatory *j* on all the other explanators, R_j^2. In other words, coefficient of determination (R^2) of a multiple regression model that estimates the relationship between one key vehicle attribute and four others is calculated, and tolerance for the vehicle attribute is obtained by $1\text{-}R^2$. Additionally, variance inflation factor (VIF) obtained by $1/R^2$ is also used to detect multicollinearity. A tolerance of less than 0.20 or 0.10 and/or a VIF of 5 or 10 and above indicates a multicollinearity problem (O'brien 2007). Table 3-6 shows tolerances and VIFs for five key vehicle attributes, and it was identified there is no multicollinearity among the five key attributes.

Thereafter, the author preliminarily estimated the customer utility function using the Limdep® program, which estimates the types of models

Table 3-6. Detection of multicollinearity among five key attributes

	Tolerance	VIF
Accelerating ability	0.81	1.23
CFE	0.72	1.39
Width distribution	0.29	3.47
Roominess	0.89	1.12
Sporty look	0.26	3.89

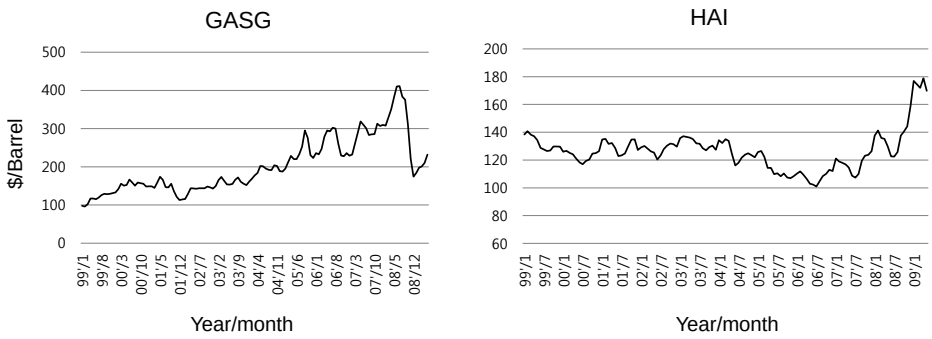


Figure 3-9. Changes of market environments

analyzed using cross-section data with linear regression (Greene 1995). Note that although vehicle price and VDS do not factor into the utility function, these variables were estimated together in order to identify their effects on consumer behavior. In the result, it was found that the p-values of the estimated preferences were too large to explain customer behaviors. This preliminary analysis drove us to find other factors affecting vehicle purchases and to adjust the customer utility function according to those factors.

3.3.2.2 Effects of environment changes on customer preferences

Consumer preferences on automobile attributes are sensitive to the market environments in which they are made (Train 2003). The mass media have reported that consumer demand in the automobile market is shifting to a new paradigm due to the rise in oil prices and the worldwide economic recession.

To capture the effects of market-environmental changes on customer preferences, the Housing affordability index (HAI), a key economic indicator, along with the oil prices in cents per gallon (GASG), a general proxy for prices,

were chosen among the several available factors. Figure 3-9 shows the fluctuation of market-environmental indices between 1999 and 2009. In order to identify which attribute preference is influenced by the HAI and/or GASG indices, a regression analysis relating each index to each attribute preference estimated with the basic model was performed. In the result, it was found that HAI influences preferences on weight-per-horsepower and wheelbase proportion, and that GASG drives preferences on those same two factors as well as combined fuel economy, width distribution, and roominess.

To estimate the effects of market environment changes on customer preferences, the author formulated the equation of the mean utility level, δ_{ij}, which added terms representing the interaction between HAI /GASG and vehicle attributes. Thus, the utility function was represented as

Table 3-7. Estimated preferences on vehicle attributes with environmental factors

variable	Coeff.	Std.Err.	t-ratio	P-value
ONE	1.167	0.646	1.807	7%
P_10000	- 0.349	0.127	-2.740	1%
Accelerating ability	0.758	0.086	8.797	0%
CFE	- 0.078	0.028	-2.807	1%
VDS	- 0.008	0.001	-15.052	0%
Roominess (normalized)	0.861	0.236	3.656	0%
Sporty look (normalized)	- 2.535	0.583	-4.350	0%
Width distribution (normalized)	- 0.135	0.095	-1.434	15%
HAI*Accelerating ability	- 0.001	0.000	-9.907	0%
HAI*Sporty look	0.013	0.004	3.795	0%
GASG*Accelerating ability	- 0.003	0.000	-8.968	0%
GASG*CFE	0.001	0.000	7.694	0%
GASG*Roominess	- 0.001	0.001	-1.488	14%
GASG*Sporty look	0.004	0.001	3.650	0%

$$U_{ij}(A_j, \xi_j, p_j; \theta) \equiv A_j\beta - \alpha p_j + \xi_j + X_j B + \varepsilon_{ij} \quad (3\text{-}3)$$

where $\delta_j \equiv A_j\beta - \alpha p_j + \xi_j + X_j B$ and *Xj* is a vector containing six attribute/environmental factor interaction terms: HAI*Accelerating ability, HAI*Sporty look, GASG*Accelerating ability, GASG*Sporty look, GASG*Combined fuel economy, and GASG*Roominess. Further, B is the coefficient of the interaction terms, and measures the extent to which an increase in an environmental factor changes coefficients on preferences. Table 3-7 provides the estimation results of the coefficients of the utility function along with the interaction terms. The oil price, GASG, adversely affects preferences on weight-per-horsepower and roominess, whereas it positively influences fuel economy and wheelbase proportion. HAI positively affects preferences on weight-per-horsepower and wheelbase proportion.

Market environments are continuously changing, and changes in customer preferences on car-model attributes followed. Therefore, the author determined the attribute preferences of customers under various scenarios. Three

Table 3-8. Changes of coefficients for different scenarios

	Scenarios		
	Scenario 1	Scenario 2	Scenario 3
Variable	(GASG,HAI)=(300, 120)	(GASG,HAI)=(250, 120)	(GASG,HAI)=(250, 140)
P_10000	-0.3490	-0.3490	-0.3490
VDS	-0.0084	-0.0084	-0.0084
Roominess	0.4292	0.5012	0.5012
Width distribution	-0.1352	-0.1352	-0.1352
Sporty look	0.1777	-0.0074	0.2597
CFE	0.1379	0.1019	0.1019
Accelerating ability	-0.2758	-0.1286	-0.1536

scenarios with sets of GASG and HAI, (GASG, HAI) = (250,120), (250,140), (300, 120) were considered. Customer preferences changed as market environmental factors were altered. Table 3-8 shows the changes of the coefficients of attributes according to the three scenarios. Because the coefficients of the interaction terms were estimated earlier, the author could determine the attribute coefficients irrespective of the scenario. Thus, the revised mean utility function was formulated as

$$\delta_j \equiv A_j(\beta + B \cdot I) \tag{3-4}$$

where I is the vector of the environmental factors affecting the relevant attributes under a given scenario.

Based on the customer utility function and all engineering design constraints, the determination of a new vehicle's specifications can be formulated as an optimization problem that finds the specification maximizing an automobile's customer utility, as shown in Table 3-9. The author accessed the optimization model in Matlab® , and used the 'fmincon' function to identify the solution. Finding the solution begins with the current model's specifications and then entails the determination of a vector maximizing the objective function (i.e., the customer utility) subject to the following engineering design constraints: min/max inequalities, interrelationship equations, and nonlinear inequality. Note that although there are several algorithms suitable for solving the optimization problem, a performance comparison is beyond the scope of this paper.

3.4 Model Validation

The determinative power of the proposed model was validated by means of comparison with the specifications of a certain previous car model and with the actual specifications of the corresponding next-generation model. A models of mid-size sedan sold on the U.S. market, Honda Accord, was chosen for the purpose of the validation. Table 3-10 compares the determination results

Table 3-9. Formularization of determination of new-vehicle specifications

GIVEN

(a) Launch year
(b) Displacement
(c) RPM for maximum horsepower
(d) Final drive ratio
(e) Transmission
(f) Brand dummy (Toyota, Honda, Nissan, EU, GM)
(g) Tirewidth
(h) Target CFE-HP technology level

FIND:

14 specifications:
OAL, OAW, OAH, Wbase, HeadFRT, HeadRR, leg FRT, legRR, WTFRT, WTRR, HP, Fuel Eco, Torque, CWT

to **MAXIMIZE**

The merchantability (M) of auto model to develop

Subject to:

Max/min inequalities:

- $-OAL + 0.28\,Displ \le -3782.0$
- $1.73OAW - Wbase \le 462.0$
- $-0.43OAL + Wbase \le 705.82$
- $0.07OAW - OAH \le -1225.0$
- $0.13OAW - TireWid \le 46.30$
- $OAL \le 5200.0$
- $0.29OAL - OAW \le -356.0$
- $-0.35OAH + HeadRR \le 468.01$
- $-0.48OAH + HeadFRT \le 338.20$
- $-0.19\,LegFRT \le -1045.0$
- $RPMHP \le 8000.0$
- $-Displ \le -1500.0$
- $-OAW + r\min \cdot Wbase \le 0$
- $OAW - r\max \le 0$

Style concept constraints:

- $OAL + 1.272OAH + 1.281Wbas \ge 0.995|x||s|$
- $OAW + 1.299OAH + 1.221WTFRT \ge 0.995|x||s|$
- $OAH + 1.126HeadFRT + 1.151HeadRR \ge 0.995|x||s|$
- $Wbase + 0.880LegFRT + 0.786LegRR \ge 0.995|x||s|$
- $OAL + 1.088OAW + 1.284WTRR \ge 0.995|x||s|$

Relationships:

- $L = CFE \times HP \times \sqrt{CWT} / Displ.$
- $CFE = CFE0 * FEInflation$
- $HP = x\_HP0 * HPInflation$
- $-HP + 6.43HondaDM + 2.46NissanDM + 0.00129Torque * RPM = 7.19$
- $0.17OAL + 0.83OAW + 11.11Torque - CWT + 1.97TireWid + 95.81EUDm - 35.09Nissan = 1530.2$
- $-CFE / FEInflation - 0.0012RPMHP - 0.31FDR + 0.51TM + 1.1Toyota + 1.20Honda + 41.57WD = -28.519$

with various market-environmental scenarios and an actual Honda Accord model. Honda launched a completely revamped model of Accord in 2012, following upon the previous 2008 model. Thus, in regarding Accord 2008 specifications as the starting point for the new model 2012, the author assumed the three market-environment change scenarios (250,120), (250,140) and (300, 120), and determined the new model's specifications for each. The mean value of the 2010-2012 environmental factors, as shown in Table 3-10, is (290,150), obtained from the Fig. 3-9 data; the closest scenarios were scenario 2 (250, 140) and scenario 3 (300, 120). The author compared the determination results for these two scenarios with the actual Accord 2012 specifications. The respective difference ratios are expressed in Table 3-10 beneath the scenarios' specification values as percentages (in parentheses).

The proposed model made highly accurate guesses with respect to the dimensional specifications, though slightly less accurate determinations of performance specifications such as maximum horsepower and combined fuel economy. According to the results of our analysis of the tendencies of consumer preferences according to the change of environmental factors, the dimensional specifications can be considered to be reasonable in the context of

Table 3-10. Comparison of results with real model – Honda Accord

	Environmental factors		Key specifications of automobile							
	GASG	HAI	OAL (mm)	OAW (mm)	Wbase (mm)	OAH (mm)	CWT (kg)	Displ. (cc)	HP (ps)	CFE (mpg)
A previous model (2008)	290	135	4930.1	1846.6	2799.1	1475.7	1489	2354	180	32.03
Scenario 1	250	120	4979 (+0.4%)	1870 (+1.4%)	2852 (+1.8%)	1476 (+0.1%)	1568 (+2.1%)	2354	207 (+15.0%)	37.4 (+2.4%)
Scenario 2	250	140	4936 (-0.5%)	1858 (+0.7%)	2833 (+1.2%)	1476 (+0.1%)	1549 (+0.9%)	2354	207 (+15.0%)	37.5 (+2.7%)
Scenario 3	300	120	4887 (-1.5%)	1844 (0.0%)	2812 (+0.4%)	1476 (+0.1%)	1531 (-0.2%)	2354	207 (+15.0%)	37.7 (+3.3%)
A new model (2012)	290	150	4960	1845	2800	1475	1535	2354	180	36.5

the oil-price increase and the level of HAI, representing the economy, after 2012. Meanwhile, there was a gap between the determined performance specifications and those of the actual model. This arose from the fact that combined fuel economy and maximum horsepower improvements fell short of our expectations.

Chapter 4. Re-architecting modular systems in incremental design

Manufacturers in various industries, in endeavouring to maintain their competitiveness, constantly modify or redesign existing products by incorporation of new technologies. Most design research has concentrated on the creation of new products; there has been relatively little work done on the redesign of existing products retaining their original architectures. This paper proposes a methodology to re-architect an existing system that has modular architecture when new technologies are to be infused via a set of new components. The methodology defines re-architecting operators through the use of a design structure matrix (DSM) technique to systemically operate pre-existing modules. According to the operators, all feasible alternative architectures are then generated. The best architecture, that which changes the original system as little as possible and has a high modularity, is determined by developed algorithms that control the operators. The proposed methodology was demonstrated with hydrogen-fueled internal combustion engines (H_2ICEs) embodying several new technologies incorporated into traditional internal combustion engines. The results showed that the existing system requires re-architecting as the new technologies were added, even though it had a high level of modularity. The new architecture for the H_2ICEs was generated by determining, with the proposed algorithm, the best re-architecting strategy available.

Keywords: Re-architecting; incremental design; product architecture; design structure matrix; change effort; modularity;

4.1. Introduction

Many recently introduced products and services in various industries are powerful exemplars of the significance of modification or redesign of existing systems to a firm's success. Eckert et al. (2012) noted that many product development projects proceed via modification of existing products rather than through *ab initio* design. Minimizing novelty to reduce risk and cost by maximizing the reuse of existing, already tested and proven components is key to design success. Indeed, in the automotive industry, hybrid vehicles, incorporating relatively mature conventional-vehicle technologies, minimize the risks incurred in the adoption of the radical technologies necessary to fill the gap that exists between conventional internal combustion engine technology and hydrogen fuel cells or other alternative fuel approaches (Friedman 2003). However, when a new product is developed as based on an existing system but with new components, changing not only the components but also the entire architecture is unavoidable.

The proposed method of re-architecting in incremental design explicitly recognizes the existing system, as the foundation of the new system, focuses on the transformation of the existing architecture into the optimal architecture of the new system. Vast amount of prior research on designing modular systems or building product platforms have proposed numerous methodologies to determine the optimal architecture for developing new products, implicitly acknowledging the existence of the previous design. Although it is imperative to determine the goal of the new architecture, the paths of transformation, from the existing to the optimal architecture for a new system, can be

exceedingly varied; furthermore, the optimal architecture itself should depend on the transformation path selected to meet the new requirements. Therefore, the method proposed in this paper models the transformation of an existing architecture by reflecting required changes. The present study determined the optimal architecture by consideration of the relevant transformation characteristics.

In this research, *incremental design* is defined as *a process of modifying or redesigning an existing system that carries over not only previous core components but also modular architectures in order to meet the required incremental changes*. In the nature of incremental design, existing modules containing core components are carried over to the new system without alteration. This reapplication of core components reduces both design costs and time. Specifically, because complex-system design incurs reliability and feasibility risks (Wyatt et al. 2009), reusing or slightly modifying existing components that have been verified through one or more product generations can alleviate or even remove such risks (Eckert et al. 2012). Henderson and Clark (1990) showed that architectural innovation that changes a product's core design concepts nullifies the usefulness of an established firm's architectural knowledge. Moreover, Sosa et al. (2004) and Tripathy and Eppinger (2011) ascertained module interfaces and corresponding team interactions are significantly aligned. Certainly, the carrying over of existing modules is integral to the efficiency of incremental-design-based new-system design.

In carrying over the core modules of the existing system, it is imperative to design a new architecture that not only effectively and efficiently reflects those new requirements but also retains the strengths of the previous architecture. In incremental design *re-architecting* is defined as *re-arranging carried-over modules and new components according to adopted incremental changes*. To determine the optimal re-architecting strategy for the new system, the present study formulated it with respect to two major criteria: 1) minimization of change effort, and 2) maximization of the modularity of the new architecture. First, re-architecting minimizes change effort applied to the existing system that arise either from adopting newly required changes or their propagation. Incremental design unavoidably generates changes to the existing system because the new system is designed based on the fundamentals of the previous system. If change efforts in incremental design outweighs the cost of clean-sheet design, incremental design loses its rationale (Eckert et al. 2012). The second major criterion of re-architecting strategy is to maximize the modularity of the new architecture. The modularity of the existing system is compromised not only by changes of existing components and their interfaces based on the new requirements but also by incorporation of new components embodying new technologies. Although a higher degree of modularity does not directly effect better product performance, many researchers have mentioned emphasized the significance of higher-level modularity to successful new-product development processes (Ulrich 1995, Baldwin and Clark 2000, Suh 2001).

The present study's specific contribution to the field is its utilization of a design structure matrix (DSM) to analyze the existing system and, then, to architect the new system. The author argues in these pages that the DSM technique enables a rigorous and systematic approach to the characterization of product architecture. Indeed, many recent studies have explored product architecture using the DSM technique (Clarkson et al. 2004, MacCormack et al. 2006, Sosa et al. 2007, Suh et al. 2010, Eppinger and Browning 2012).

4.2. Methodology: Re-architecting through incremental design

The present study proposes a methodology to re-architect an existing system that has modular architecture when new technologies are to be infused via a set of new components. Figure 4-1 is a schematic diagram of the proposed model. The core modules of the existing system, which constitute the baseline of the incremental design, are included in the proposed methodology in conjunction with the adopted incremental changes to the modules' respective components; new components embodying new physical-domain technologies are included as well. The existing system and the new components, in the re-architecting methodology, are represented as DSM, namely "revised DSM", which simply expresses the architecture of the existing system and the required changes. Thereafter the methodology generates, for the new system, all possible alternative architectures that can be configured with the carried-over modules and new components to accommodate the new requirements with

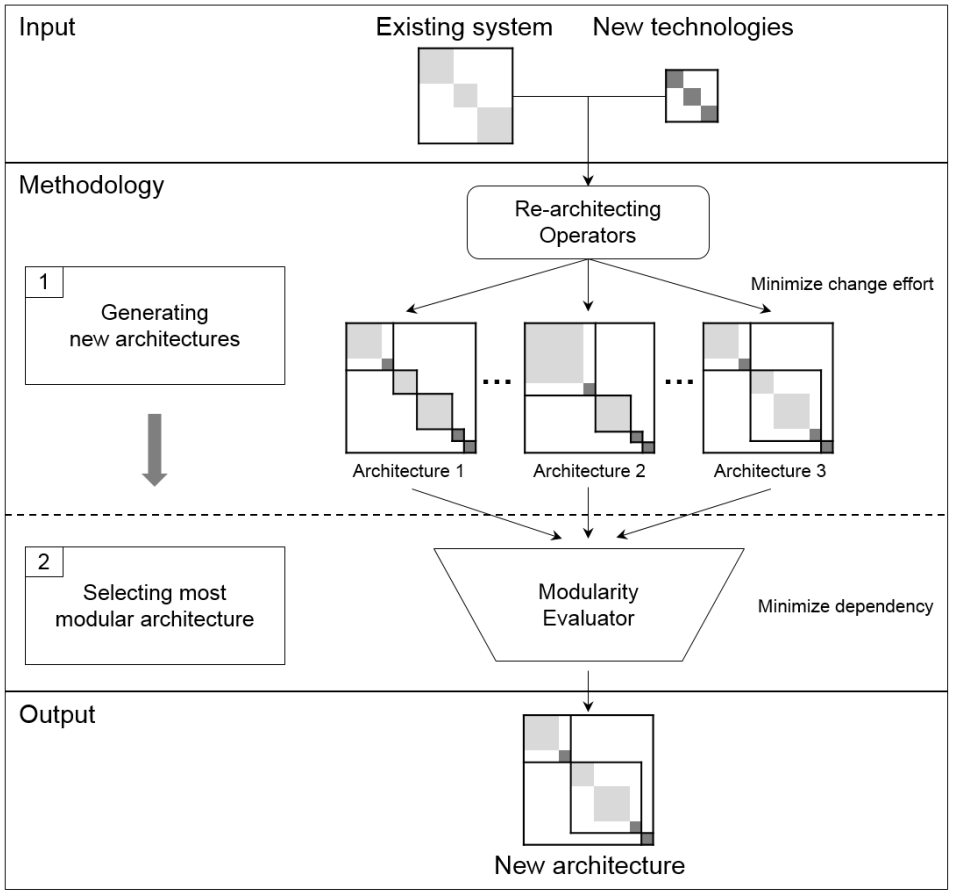


Figure 4-1. Re-architecting methodology

minimal changes to the existing system. The generation mechanism is systemically represented by *re-architecting operators* with algorithms for the operator's activation. Subsequently, the methodology evaluates the generated architectures and selects that which has the maximum modularity among modules.

Our re-architecting methodology is based on the following assumptions. First, the existing system has modular architecture that includes one to one mapping from functional elements in the function structure to modules

composed with the components of the product and specifies decoupled interfaces between the modules (Ulrich 1995). Second, carried-over modules cannot be split into submodules, because in incremental design, the new system inherits the core modules of the original design. Third, the carried-over modules have de-coupled interfaces, as the design of the core modules, having been stabilized through several generations, approach the ideal. As such, a change in any module will not propagate to other modules (Ulrich 1995). Fourth, in order to add new components to a particular existing module, their interfaces must be standardized in relation to the other, surrounding modules. This assumption accords with the second assumption that the standardized interfaces of a module stop propagating changes to the other modules.

4.2.1. Re-architecting Operators in Incremental Design

Modular design offers dynamic possibilities inherent in the several ways in which modules can be re-arranged and structures altered. Baldwin and Clark (2000), in considering those possibilities, proposed the following six modular operators. Adopting and modifying this concept, the author defined the following five *re-architecting operators*: augmenting, excluding, merging, porting, and substituting, which span the altered architecture in incremental design.

The author uses the DSM technique to define the re-architecting operators and understand the mechanisms by which they change the architecture. How each re-architecting operator works in the DSM technique and what sequences for the existing system are accompanied through microscopic viewpoint. To explain the re-architecting operators, the author begins by looking at

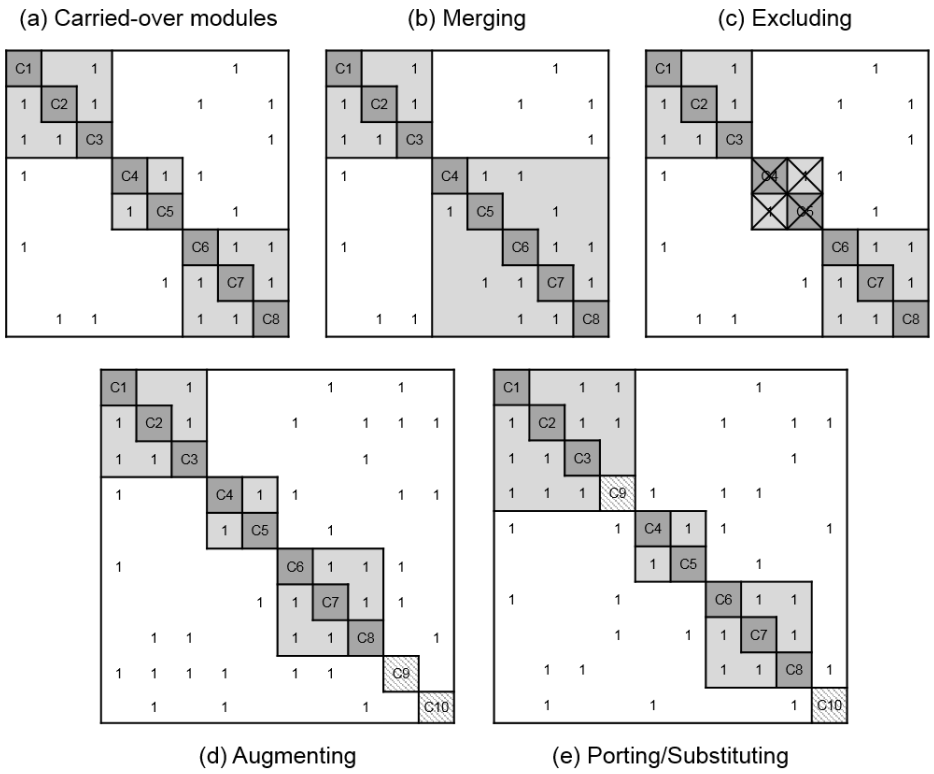


Figure 4-2. Re-architecting operators

the generic modular design of an existing system with a DSM. In our example, Figure 4-2(a) represents the DSM of a modular product that has three core modules composed of eight components (the modules are represented by the three shadowed blocks). Each module has its own components, and the components interact with other components, in the same module or another module, through several types of interfaces. If two components interact through an interface, the cross-sectional cell is labeled "1". The five re-architecting operators are defined as follows.

Augmenting is defined as the adding a new component or submodule in order to add new technologies to the existing system. In Figure 4-2(d), a new module comprising the components C9 and C10 is added to the existing system. Component C9 affects the components of the three original modules through the relevant interfaces, which are represented in the DSM as a column vector of C9.

Excluding is defined as the removal of a module that is not needed in a new system. If the existing modular system is broader than the new system, the designer can select a subset of existing modules for exclusion. Consequently, not only the selected modules but also all of the pertinent interfaces are terminated. Figure 4-2(c) shows the results of excluding from the existing system a module composed of components C4 and C5.

Merging is defined as the amalgamation of existing modules into a new submodule, as represented in Figure 4-2(b). In a modular architecture, each existing module has its own "shell" that isolates its interior components from the surrounding system. Once the modules are merged, a new isolation shell, covering all of the components of the merged modules and the interfaces existing among them, is generated. In practice, it is difficult to merge two different modules. However, in theory, it could be hypothesized that two ideal modules could be merged without incurring any large increase in cost in that the carried-over modules of the existing system have created standardized interfaces throughout their long development history.

Porting is the migration of a component from a module (a new component in the incremental design) to an existing module. As a consequence of

porting a new component to the selected module, changes occur that are propagated to the related components within the selected module.

Substituting, occurring when the existing module needs changes to reflect new requirements, is defined as replacing an existing module with its upgraded module that adds new components or updates the original components/interfaces. Figure 4-2(e) shows the result of the substitution of a new module, which includes the new component C9, for an original module constituted of C1, C2, and C3 components.

4.2.2. Determining optimal re-architecting strategy in incremental des ign

The re-architecting methodology generates all possible alternative architectures spanning the design space in which the potential solution is to be found for the new system. Within the design space, the proposed methodology determines the optimal architecture for a new system by considering the following two major criteria for re-architecting in incremental design: 1) the existing system is changed as little as possible, and 2) a high level of modularity is retained. Those major criteria are included in the core algorithm that activates the re-architecting operators.

In this study, the re-architecting methodology manipulates the existing system's architecture with the re-architecting operators through utilizing the DSM technique. The basement for generating alternative architectures is to build the revised DSM that reflects the required changes to the existing system by applying the re-architecting operators, such as augmenting, excluding, and

substituting. Herein, not only what module of the existing system should be excluded or substituted by its upgraded version but also what should be augmented, is determined and applied. Therefore, the revised DSM is a "base board" on which the designers span the new architecture through iterative application of the re-architecting operators subject to the two re-architecting objectives. The following sections describe in detail the algorithms required for the three stages of new architecture generation and selection.

4.2.2.1 Definition of modular-architectural space for existing system

The re-architecting methodology first spans and defines the modular-architectural space by determining all possible combinations of merged carried-over modules. As the existing system accommodates the required changes, the modularity of the baseline product will be compromised. Hence, even though the baseline (original) product has an ideal modular architecture, it is necessary to explore new, higher-modularity boundaries for the new system. In the modular-architectural space, each point represents a new system composition of feasible merged modules, which are made up of the carried-over modules. Additionally, the overall modular-architectural space contains all alternatives for new components to explore and identify where they should be ported to minimize change effort in entire system.

To generate feasible merged modules with the carried-over modules, the author changed our way of thinking. In other words, the question of how to merge modules can be reversed, and the question of how to split the entire

modular system can instead be asked. Based on the concept of disassembly used in a previous study (Suzuki et al. 1993, Lambert 2002, Kwak et al. 2009), the author defines *splitting* as an action that decomposes a parent submodule into two child submodules while standardizing all of the interfaces among the relevant subsystems. With this splitting action, all possible merged modules can be generated based on the predefined modules of the existing system.

Splitting the existing system is characterized by the actions and action vector of the resulting modules. To represent the relationship between action and generated modules, the author defines a *decomposition matrix* T of size $M \times N$, which has M feasible modules and N candidate actions. An element T_{mn} equals -1 if action *n* decouples a parent module *m*, and +1 if the action generates a child module *m*. Table 4-1 shows the decomposition matrix

Table 4-1. The decomposition matrix

	A0	A1	A2	A3	A4	A5	A6
ABC	1	-1	-1	-1	0	0	0
AB	0	0	1	0	-1	0	0
BC	0	0	0	1	0	-1	0
AC	0	1	0	0	0	0	-1
B	0	1	0	0	1	1	0
A	0	0	0	1	1	0	1
C	0	0	1	0	0	1	1

of a particular system with three existing modules. For instance, action 1, *A1*, splits parent module *ABC* into submodule *AC* and submodule *B*, and action *A6* breaks down submodule *AC* to *A* and *C*. Here, several combinations of actions generate different types of granularity as a result of splitting. To model the combination of actions, the splitting action vector *X* of size $N \times 1$ is defined. An element x_n is a binary variable that equals 1 if action *n* is conducted. Therefore, depending on the composition of *X*, the generated modules are identified as follows in Equation (4-1):

$$S = T \cdot X \qquad (4\text{-}1)$$

where *S* is the generated module vector of size $M \times 1$, and element s_m is either +1 if *m* modules are created or 0 if not. To be feasible, the splitting action vector *X* must make all of the elements of vector *S* satisfy $s_m \geq 0$. All feasible action vectors can be enumerated by identifying whether candidate *X* satisfies this constraint or not. Figure 4-3 shows our example of how the merged module is generated with the existing modules. When a particular system has three modules, the splitting action vector $X = [1\ 0\ 0\ 1\ 0\ 0\ 0]'$ generates the module vector $S = [0\ 0\ 1\ 0\ 0\ 1\ 0]$, which represents module *A* and the merged module *BC*, by the inner product $T \cdot X$ with the decomposition matrix represented in Table 4-1. In this way, once designers determine the vector *X*, split modules are generated from the existing system. Consequently, the merged carried-over module set *S* is determined according to the composition of an action vector *X*. Specifically, this module set *S* represents the boundaries

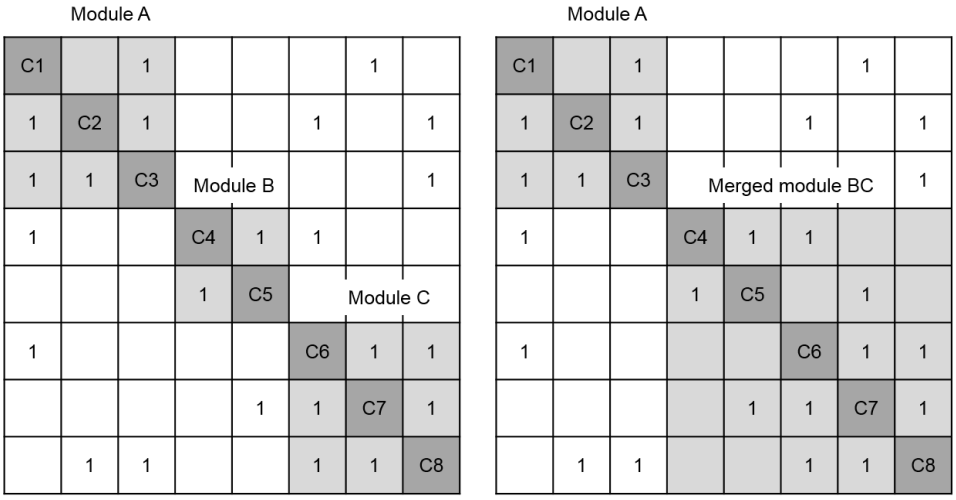


Figure 4-3. Generation of merged module from carried-over modules

of the newly merged modules of the existing system, and becomes a point in the modular-architectural space for the existing system. Additionally, it becomes the baseline in reference to which new components are incorporated, as described below.

4.2.2.2 Exploration of extended space for incorporation of new components

Our re-architecting methodology extends the modular-architectural space of the existing system by incorporating new components to the carried-over module architecture, defined in the pre-extend space. Thereby, the extended space generates all possible alternative architectures spanning the design space in which the potential solution of re-architecting is to be found; each point represents a new architecture that ports new components into the carried-over

merged modules of the existing system. In the extended space, the methodology explores new architectures that minimize the change effort incurred in incorporating new components into the existing system.

As an existing system adopts new components, the incremental changes from those new components affect the related components, and thus the interfaces among components propagate changes through the entire system. Therefore, it is imperative to understand change propagation in modular architectures and to establish, on that basis, an architecting strategy for effective porting of new components. Indeed, several studies have devised metrics for assessment of design-change impacts. Clarkson et al. (2004) identified a combined impact that takes into account both direct and indirect change propagation paths. Suh (2005) introduced the change propagation index (CPI) to measure an element's degree of change propagation for a given change. Martin and Ishii (2002) developed indices to understand the changes and their effects on components in various generational products. Additionally, MacCormack et al. (2006) characterized the structure of product design as measuring the propagation cost, which is an instance for the degree of coupling. Based on these studies, the author examined how new components generate changes on an existing system and how the changes propagate in a modular architecture.

The changes induced by porting a new component into the existing system are modelled in the form of Figure 4-4's simple example. The left side of the figure represents new components, C9 and C10, entering the existing system composed of three modules, using the DSM technique. Specifically, each component *i* interacts with another *j*, which interaction is characterized

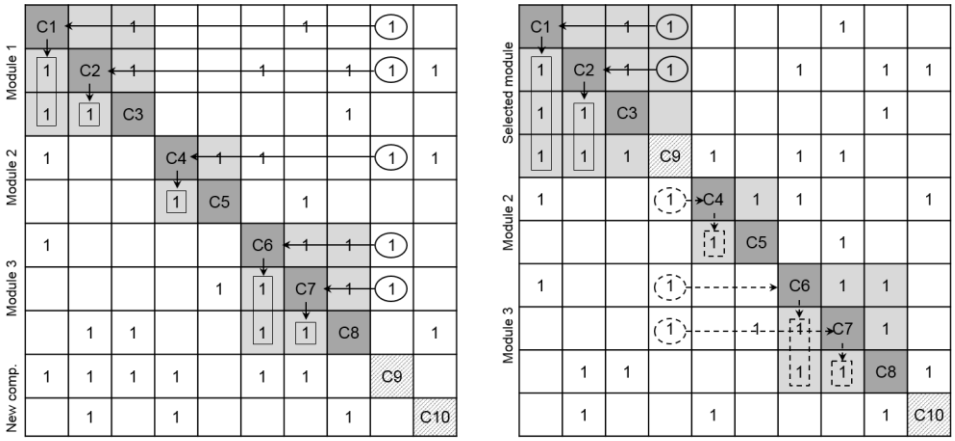


Figure 4-4. Induced changes due to incorporation of new components

by the existence of interdependency between them. As previously noted, augmenting a new component effects changes to existing modules through the interfaces. The seed components that are affected by a new component are scattered to their original modules, whereupon they propagate changes to the interlinked components within their modules' boundaries. On the basis of the assumption of re-architecting in incremental design, the carried-over modules of the existing system have interfaces that are de-coupled from each other, because the design of the core modules has been stabilized through several generations to approach the ideal. In a modular architecture, a change in a module does not propagate to other modules in order to work correctly because the modular architecture includes, as a standard feature, de-coupled interfaces between the modules (Ulrich 1995). Meanwhile, a module's components include coupled interfaces, whereby a change triggered by a new component propagates within the module. Therefore, a new component propagates

changes to carried-over components because it does not include standardized interfaces. However, the generated changes are kept within the carried-over modules. Therefore, the induced change in the merged carried-over module $s \in S$ composed of j components takes, due to new component i, the form

$$Induced\ change(i \rightarrow j \,|\, \text{in exsting module } s) = \sum_{J} \alpha_{i,j} \cdot I_{j,i} \cdot R_j \quad (4\text{-}2)$$

where $I_{j,i}$ is a binary variable, marked in the DSM(j,i), that indicates the interdependency between the new component i and the existing component j; the merged carried-over module set $S = \{s_1, ..., s_n\}$ is a particular point in the modular-architectural space, defined by Equation (4-1); $\alpha_{i,j}$ is the change density, and R_j is the number of change propagation routes due to the existing component j.

The structure of each module is characterized by analysing the propagation routes through which a change to any single component causes a direct change to other components in the module. R_j can be represented as

$$Propagtion\ \ Routes(j \rightarrow k \,|\, \text{in the module}) = \sum_{K} I_{k,j} \quad (4\text{-}3)$$

where $I_{k,j}$ is a binary variable, marked in the DSM (k, j), that indicates interdependency between change receiver j and its interlinked component k in the module. Note that the proposed methodology only considers direct change impacts; nonetheless, it can be expanded by taking into consideration the indirect

propagations addressed in other studies (Clarkson et al. 2004, MacCormack et al. 2006).

As an example, Figure 4-4 (right) shows the porting of a new component and substituting an existing module with the new module incorporating the new component. In this case, component C9 is ported to the carried-over module 1 and generates changes to component C1 and component C2. C1 in turn has direct impacts on components C2, C3, and C9, and similarly, C2 has direct impacts on components C3 and C9. Therefore, C1 has three propagation routes, $R_{C1} = 3$, and C2 has two, $R_{C2} = 2$, which are marked with rectangles. When a new component is ported to a particular module, the boundary of the selected module isolates its change propagation to the surrounding components. Hence, porting a new component to a particular module reduces the amount of change-related effort required across the entire system. The amount of change reduction achieved can be represented as $\sum_{S \setminus g} \sum_{J} \alpha_{i,j} \cdot I_{j,i} \cdot R_j$, where new component i has impacts on component j in the merged carried-over module g, which isolates change propagation to the surrounding modules $S \setminus g = \{s \in S \mid s \neq g\}$. In Figure 4-4 (right), the change reductions are represented by dotted lines.

Porting a new component into the selected module incurs a cost for standardizing the interfaces related to the surrounding modules, as defined in the re-architecting operators. As the existing modules have already de-coupled the changes from the other components, adding a new component should standardize the interfaces that exist outside of the module to enter; in other

words, the new components' interfaces are originally not standardized for all of the existing modules. The porting cost of a new component i can be represented as $\sum_{J} t_{i,j} \cdot I_{i,j}$, where $t_{i,j}$ is the effort to standardize the interface for the interdependency $I_{j,i}$ outside of the selected module. Finally, the induced change effort necessary for porting of a new component i into the carried-over module g is

$$\sum_{S}\sum_{J} \alpha_{i,j} \cdot I_{j,i} \cdot R_j - \sum_{S \setminus g}\sum_{J} \alpha_{i,j} \cdot I_{j,i} \cdot R_J + \sum_{\text{Out of g}} t_{i,j} \cdot I_{j,i} \tag{4-4}$$

where $S = \{s_1, ..., s_n\}$ is the merged carried-over module set, which is a particular point in the modular-architectural space, defined by Equation (4-1), and j is marked for the existing components that receive changes from new component i within the module S. The first term represents the induced changes of the overall carried-over modules of the existing system, the second term is the amount of change reduction achieved by porting the new component into a selected module g, and the last term computes the standardization effort necessary for a new component to be added to the selected module.

The proposed algorithm attempts to determine the new components' optimal allocation to the given merged carried-over module set *S*, which is a point in the modular-architectural space of the existing system, such that the change effort is minimized. The challenge is that, for all of the possible porting combinations (which new components is ported to which module), the solution space becomes too large to search. Therefore, an iterative search process

is adopted, whereby a randomly picked component is the "migrator" from the new components that induces changes through the interdependencies. The migrator compares all of the candidate carried-over modules with the respective induced change efforts represented in Equation (4-4), and selects the module that yields the greatest marginal reduction. If there are no negative marginal changes, the element remains in its current location. When there is a module that creates a marginal reduction, the migrator is ported, and the existing module is substituted with a new module that adopts the migrating component. The porting and substituting process continues in an iterative manner, stopping when there is no improvement in cost that exceeds a predefined threshold.

4.2.2.3 Selection of architecture with minimum dependencies

The re-architecting methodology finally selects the most modular architecture that has minimum dependencies among the alternatives in the extended architecture space, as illustrated in Figure 4-1. In the previous steps, the re-architecting operators concentrate on producing various alternative architectures that minimize the change effort for the existing system and surely bring about the destruction of the modularity of the existing system, which had been stabilized through its long development history. Therefore, it is imperative not only to minimize change effort but also to secure the new system's modularity. In the present study, the *modularity evaluator* was devised to evaluate the modularity of the generated architectures. This proposed evaluation method is based on prior research that identified modules of complex systems (Whitfield et al. 2002, MacCormack et al. 2006, Yu et al. 2007b). The present paper

herein posits *modularity* as *the degree of grouping that puts tightly connected components in the same boundary with no dependencies between these boundaries*. To measure the modularity of the entire system, the author models a dependency between elements i and j, in a manner similar to that of MacCormack et al. (2006), as

$$Dependency(i \to j \,|\, \text{in same module}) = I_{j,i} \cdot n^{\lambda} \tag{4-5}$$

$$Dependency(i \to j \,|\, \text{not in same module}) = I_{j,i} \cdot N^{\lambda} \tag{4-6}$$

where $I_{j,i}$ is a binary variable indicating the interdependency between components i and j; n is the number of components in the module that i and j are located within; N is the total number of components of the new system, the revised DSM size and, λ is a setting parameter. These equations postulate that dependencies between components within the same module incur a cost that is lower than that of components belonging to different modules. The alternative generated architectures have different module boundaries; as a result, depending on the alternative, a particular set of two components can either be within the same module or belong to different modules. The modularity evaluator calculates the modularity of an alternative with Equations (4-5) and (4-6) as reference boundaries of the participating modules and the dependencies of the components; all of the dependencies among the components are summed according to where each dependency is located. Consequently,

the proposed modularity evaluator will select, from among the generated architectures in the extended modular-architectural space, the best alternative architecture with the minimum dependency for an entire system. The selected architecture satisfies the following two main objectives: incurring as little change as possible, and retaining modularity.

This study focused on the mechanism for re-architecting rather than the efficiency of determining the optimal solution for a new architecture. Therefore, the proposed methodology finds an optimal solution simply by searching all possible alternatives and selecting the best one. However, the author should not that in practice, the method shows limitations as the problem becomes large and difficult. It is vital, therefore, to develop an optimal algorithm for running the re-architecting methodology in polynomial time. This issue will be considered in future research.

4.3. Case study: Hydrogen-fueled internal combustion engines

The present study had two aims. The first was to formulate a re-architecting strategy by which an existing system can be changed as little as possible while maintaining the architecture's modularity. The second was to establish a new architecture for an incremental design as a result of the formulated re-architecting strategy. These aims led designers to the choice of the imaginable new architectures that the author explores: Hydrogen-fueled internal combustion engine.

Hydrogen-fueled internal combustion engines (H_2ICEs) operate as clean and efficient power plants for automobiles and are widely regarded as the ideal engine of the future. Given that governments are putting strategic plans in motion to decrease primary energy use, the interest in H_2ICEs has been growing. The most compelling reason for these engines to be seen as ideal is the ever-increasing likelihood that they will serve as the traditional hydrogen powertrain during the initial development of the hydrogen economy. This view comes from the fact that H_2ICEs can utilize manufacturing infrastructure already developed for internal combustion engines and serve as an option to fill the gap between conventional technology and hydrogen fuel cells that are undergoing continuous development (White et al. 2006). Therefore, the author examined the H_2ICEs as a subject for incremental design that is developed based on the traditional internal combustion engines.

To examine the proposed methodology, the author used the DSM data of the internal combustion engines (ICEs) published by Smaling and De Weck (2007) and simplified the four types of connections between components to an interdependency $I_{i,j}$, which is a binary variable. Figure 4-5 shows the DSM of the ICEs as an existing system. The system has modular architecture composed of the following four modules: engine (lower/upper ends), induction system, fuel system, and exhaust system. The existing engine system has well

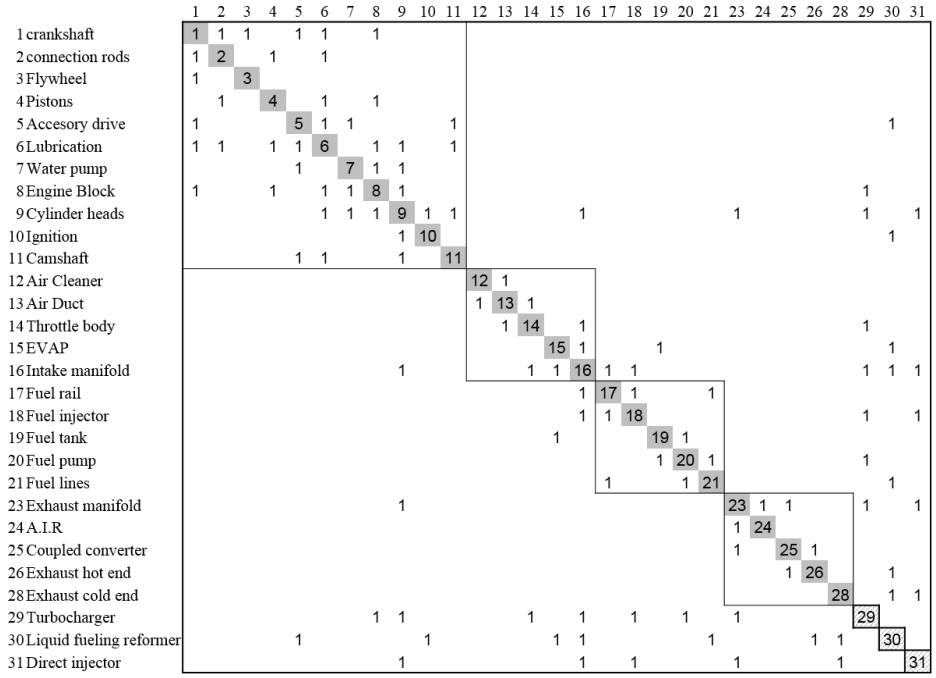


Figure 4-6. Revised DSM reflecting required changes

defined modules in that there are very few dependencies among them, as evidenced by the fact that much of the DSM outside of the modules is "white space".

H_2ICEs have several new requirements when starting from the basis of traditional ICEs. New technologies have been developed that focus on power density, NO_x emissions, and thermal efficiency. The author postulates an advanced reciprocatin g engine concept that encompasses the following advanced technology options in the near-term (Peschka et al. 1992, Jaura et al. 2004, Rottengruber et al. 2004, White et al. 2006): turbocharger, liquid fueling reformer, and direct injector. Figure 4-6 shows the revised DSM, and the input

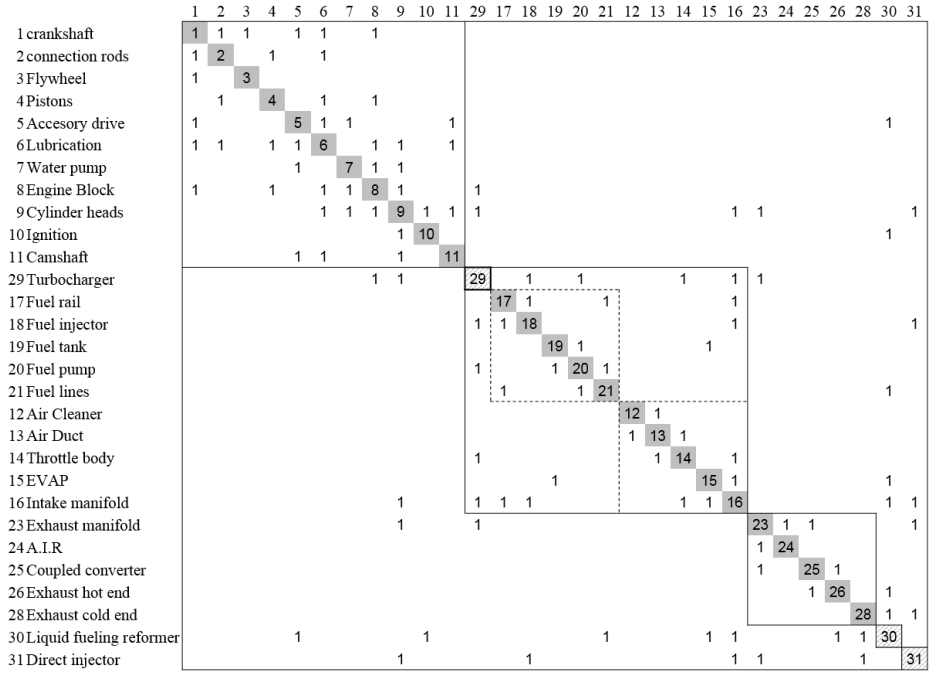


Figure 4-7. New architecture for H_2ICEs

data to the re-architecting method illustrated in Figure 4-1, which postulates the H_2ICE concept. To compose a new system of H_2ICEs, four core modules of the internal combustion engines, which include 26 core components, are carried over, and three new components, components 29 to 31 in Figure 4-6, are augmented. The prior enumerated new technologies are embodied in physical components and augmented to the ICE$_s$. Additionally, the baseline exhaust module is substituted with a new module that removes unnecessary components, such as the EGR system and the underbody converter, due to the incorporation of new technologies.

The re-architecting methodology was programmed with Matlab® . In the first step, the programmed code establishes the decomposition matrix T , which references the number of carried-over modules to the new system. In the second step, the code generates a feasible merged modules vector S that corresponds to an action vector X based on the decomposition matrix. In the third step, the existing system is augmented with the new components and the code determines whether each new component is ported to the merged carried-over modules to minimize change effort. To generate all possible alternative architectures for the new system, the second and third steps are conducted in an iterative manner for all feasible action vectors. Finally, the code assesses the modularity of all of the generated alternatives and selects the best one, which is defined as that which has the minimum dependencies over the entire system.

Table 4-2. Determined re-architecting strategy

Re-architecting operators	Object modules / components
Augmenting	Turbocharger, Liquid fueling reformer, Direct injector
Excluding	EGR system, Underbody converter
Merging	Induction and Fueling modules
Porting	Turbocharger
Substituting	Exhaust module

The revised DSM of the H_2ICEs, illustrated in Figure 4-6, is input into the re-architecting model. As the traditional engine has four modules, a decomposition matrix *T* of size 15×23 is generated; this means that 15 feasible merged modules can be generated by combining 23 feasible actions for the four carried-over modules. Based on this information, the proposed model generates all possible feasible actions, defined by Equation (4-1) as set *X* and merged module vector S of size 15×1. The three new components for the H_2ICEs are augmented and, for each merged vector, arranged to minimize change efforts.

Figure 4-7 shows a new architecture for an H_2ICE concept determined by the re-architecting model. When Figure 4-7 is compared with Figure 4-6, the two carried-over modules of induction and fueling are merged into one, and then the turbocharger is ported to the merged module to minimize the change effort. Therefore, the required changes due to the turbocharger component are isolated within the boundary of the newly generated module. Note that the carried-over modules that are merged to the new module are represented with a dotted line. Meanwhile, liquid fueling and a direct injector might remain without porting any existing module, because these new components cause no reduction in the change effort through their porting into any existing modules. Furthermore, it is apparent that the re-architected DSM of Figure 4-7 has more "white space" when compared with the unstructured architecture of Figure 4-6. Table 4-2 summarizes the re-architecting strategy that generates the best new architecture from among a large number of alternatives. There-

fore, the newly generated architecture seems to be more efficient from the perspective of modularity. Note that a different architecture can be generated by setting the predefined coefficients using real data from the automotive industry. In this case study, the author assumed that $\alpha_{i,j} = 1$ and $s_{i,j} = 1.3$.

Table 4-3 provides quantitative data comparing the architectures before and after re-architecting the H_2ICEs system. The architectures are represented in Figures 4-6 and 4-7. The author first notes that the new architecture presented by the proposed methodology requires fewer changes for the new system. Specifically, the new architecture requires 46 changes to add new components to the existing system. By contrast, the H_2ICEs system without re-architecting requires 56 changes, which represent a more than 20% increase in the required changes. This would imply that the re-architected system

Table 4-3. Comparison of H_2ICEs before and after re-architecting

	Hydrogen-fueled internal combustion engines	
Carried-over components	26	
New components	3	
	Non-re-architected	Re-architected
Required Changes	56	46
Dependencies	37,119,397	27,635,908

* The dependency results the author report uses $\lambda = 4$. It is necessary for the analyser to decide λ to some value while considering the characteristics of the object system and the sensitivity of the value.

makes changes that propagate much less than in the non-re-architected one. It should be noted that the change effort for the new system can be reduced by the choice of how to architect the carried-over modules and new components. Furthermore, the re-architecting methodology establishes a highly modular system that is effective in changing the product and generating variety. The dependencies of the new architecture are approximately 70% of those of the unstructured architecture. This dependency change implies, notably, that the new architecture is more modular than the unstructured one.

Four different types of interactions among the components can be displayed in the DSM (Pimmler and Eppinger 1994): (1) associations of physical space and alignment; (2) associations of energy exchange; (3) associations of information exchange; and (4) associations of materials exchange. These four generic interaction types are defined as follows:

Spatial: A spatial-type interaction identifies needs for adjacency or orientation between two elements.

Energy: An energy-type interaction identifies needs for energy transfer between two elements.

Information: An information-type interaction identifies needs for information or signal exchange between two elements.

Material: A material-type interaction identifies needs for materials exchange between two elements.

Clustering the DSM based on any one of these four types of interactions alone is likely to suggest a different set of clusters than clustering on the combination of interactions. This depends on the question of whether certain types of inter-

actions should be chosen or weighted more heavily than others by the objective function. If designers want to determine modules that have minimum spatial interactions on each other, modular architecture should be identified based on the spatial-type interactions. Generally, spatial proximity interactions might be more difficult to achieve via a standard interface than information flow interactions, which might be more amendable to a standard protocol. Meanwhile, multiple types of interactions can be considered by combining the interactions with the appropriated weights. Note that the present paper does not distinguish the types of interactions between two components. Nonetheless, the proposed methodology could be applied to all types of interactions. The designers or analyst can often gain useful insights by comparing the different optimal solutions (modular architecture) found when considering the different types of interactions collectively and separately.

The question that can arises given these results is whether the high modularity generated by merging the carried-over modules has an effect on the performance of the new system or not. Ulrich (1995) noted the potential trade-off that exists between product performance and modularity. However, the trade-off might not be a necessity in the case of a product that has a design that has been continuously improved based on the previous design. Indeed, MacCormack et al. (2006) showed that the evolved product, Linux, has a higher modularity than the original product, Mozilla, while its performance was improved. In the case of H_2ICEs, because internal combustion engines have evolved and stabilized in terms of modular architecture and performance,

each carried-over module, which is merged into larger modules, would not reduce the performance of the new system.

The proposed re-architecting methodology aims for the existing system to adopt innovative changes in incremental manner. Despite how innovative the added new components are, the present methodology tries to manipulate the innovation with minimizing affection to the well-established existing system. While design research has concentrated on creativity in the early stage of design and on creativity in very open-ended design tasks, in practice many design projects concern the modification or incremental development of existing systems to meet new needs and restrictions. Some new product development project, such as electronic devices, are naturally conducted from clean sheet to deliver completely new functions to customers. However, in reliability-critical industries, such as automobile or aerospace, innovation is only appealing to customers if it adds significant new functionality to the existing (not to the clean sheet); otherwise, they want confidence in the long-term performance of the product, as novelty entails risks of late delivery or product failure. Furthermore, manufacturers also does not want to cope with risks arisen throughout product life cycle. In engineering, designing complex products is therefore typically carried out by modification from existing designs, specifying changes to one or more starting designs. If designers embrace the nature of innovation in complex system, the proposed method can help them meet new requirements in lower risk level and design costs.

Re-architecting methodology can be dynamically utilized to platform strategy to the management of their product architectures in the situation of

continuously introducing new technologies. In industries characterized by a dominant product design, such as automobile, aerospace, strict interface management has to be applied in order to benefit from economies of scale and outsourcing potentials. The leading companies are developing new technologies, such as turbocharger, direct injector in automobile industry, to meet changed requirements of customers and regulators. The new technologies should be infused to the strictly established module, and the interfaces should be managed with consideration of the reusability of the module. The newly defined modules become the elements of platform for commonality strategy. The classical trade-off between optimizing manufacturing costs through integrated design versus optimizing life-cycle costs through modular design will shift toward the latter one (Meyer and Lehnerd 1997, Ulrich and Eppinger 2007). The reusability of modules for product variants can lead to significantly lower life cycle costs and makes economies of scale and scope, maintenance synergies, and improved product quality. The re-architecting methods provides new modular architectures, which actively incorporate new technologies to the existing system, to be utilized for platform-based thinking.

4.4. Summary

In this paper, a new methodology for re-architecting an existing system through accommodating newly required changes is introduced; additionally, the methodology is demonstrated using the example of hydrogen-fueled internal combustion engines. If the existing system is comprised of well-defined

modules, incremental design should not only change it as little as possible but should also retain the modular architecture. The proposed methodology defines re-architecting operators with DSM to systemically address the modules in the process of architecting. Each operator, as applied to the existing modules, generates a large number of alternatives for the new architecture. To determine the optimal architecture among those alternative architectures, the proposed methodology provides algorithms for the operator's activation. These algorithms are formulated with Matlab® and applied to a new engine concept wherein three new technologies were incorporated into traditional internal combustion engines. The results of this case study satisfactorily demonstrated the necessity of re-architecting in incremental design, even though the existing system had a well-defined modular architecture.

Chapter 5. A genetic algorithm for re-architecting in incremental design

Recently identifying optimal solution in engineering design, researchers are often confronted by difficulties arising from complex system. In principle, the optimal solution to such problems can be found by even enumeration or simple heuristics; however, in practice it is frequently impossible where the number of feasible solutions can be extremely high. Re-architecting modular systems in incremental design has such difficulties; especially for practical problem of realistic size, the explosive increase of feasible alternatives compared becomes the most challenge. In this paper, a genetic algorithm is proposed in order to intelligently search feasible solutions rather than enumerating and comparing. Solution encoder, a heuristic feasibility operator, fitness evaluator, and select/replacement model are tailed specifically for re-architecting modular systems in incremental design. The present algorithm was demonstrated with hydrogen-fueled internal combustion engines (H_2ICEs), which are developed based on the traditional internal combustion engines, as a subject for incremental design. The results showed that the proposed genetic algorithm is to make a good balance between exploration and exploitation of the search space.

Keywords: genetic algorithm; re-architecting; incremental design; product architecture; genetic operator; design structure matrix; modularity

5.1. Introduction

Incrementally designing new system by maximizing the reuse of existing, already tested and proven components is key to manage risk and costs in new product development. Eckert et al. (2012) noted that many product development projects proceed via modification of existing products rather than through *ab initio* design. The author defined *incremental design* as *a modification or redesign activity of the existing system through carrying over not only previous core components but also the modular architecture to meet the required incremental changes* (Kang and Hong 2013). The necessary condition of incremental design is that the new system carries over the existing core modules, which are composed of element components, to retain reliability and completeness of the previous product, but also to reduce development and production costs.

When a new system is developed as based on the fundamentals of the previous system but with minimal number of new components, changing not only the existing components but also the entire architecture is unavoidable. New technologies or components cause design changes to the existing system (Suh et al. 2010), and those changes propagate to the entire system through interlinks between the element components (Clarkson et al. 2004, Giffin et al. 2009). In addition, the modularity of the existing system will be broken due not only to changes of the existing system and their interfaces based on the new requirements but also due to the insertion of new components, embodying new technologies. Therefore, it is imperative to architect a new system in order that the new architecture not only effectively and efficiently incorporates new

components to the carried-over modules but also retains the strengths of the previous architecture. In this research, *re-architecting* defined as *re-arranging carried-over modules (parts) and new components according to adopted in incremental design.* Particularly in incremental design, the carried-over modules, which cannot be split into submodules, because in incremental design, the new system inherits the core modules of the original design, can become a type of constraints on new system development.

Re-architecting modular systems in incremental design is the strategy-establishment problem how to re-arrange carried-over modules with decision of whether they embrace the new components in their module boundary or not. The problem pursues two objectives: 1) minimizing change effort on the new system, and 2) maximizing the modularity of the new architecture. Indeed, new module composition through re-architecting should be able to manage the design changes by insulating their propagation within the module. Moreover, appropriately incorporating new components into the highly related module can endure the architectural strength of the previous system without impairment in modularity. Therefore, based on understanding not only the existing system's modular architecture, characterized with composed elements (components) and their interdependency, but also effects due to incorporation of new components, the re-architecting strategy can provide the efficient and effective paths of transformation, from the existing to the optimal architecture for a new system.

In the previous research of Kang and Hong (2013), the proposed meth-

odology generates all possible transformation paths of the carried-over architecture and identifies the most effective/efficient path among them. The generation mechanism is systemically described with re-architecting operators, which reconstitute modules with the carried-over modules and newly incorporated components by using the design structure matrix (DSM). Afterwards, the methodology evaluates the generated architectures and selects the best architecture, which has a minimum changes to the existing system and a maximum modularity.

Difficulties can be arisen in the proposed methodology as the modular system becomes complicated. The previous re-architecting methodology finds a solution through comparing all possible alternative architectures. Although, in principle, the optimal solution can be found by simple enumeration, in practice it can be impossible, especially for complex systems that are composed of a large number of modules, where the number of feasible solutions can be extremely high. Indeed, the core mechanism of generating feasible alternatives in the previous methodology combines how to compose feasible modules with the carried-over modules and how to port each new component into them. When the author considers a composition policy $i \in I$ that generates n_i numbers of feasible modules with the carried-over modules, each new component $k \in K$ has n_i options of entering to the generated modules. Therefore, all the newly added components have $n_i^{|K|}$ combinations of simultaneously being ported to the merged modules of the existing system. Eventually, for the entire policies I, all possible alternative architectures can be generated

as the number of $\sum_{i}^{I} n_i^{|K|}$, and to find an optimal solution (architecture), all the generated alternatives should be evaluated in terms of the induced change efforts and the modularity of the new system. As the modular systems get bigger, the number of generated feasible modules n_i brings out the exponential increase of the number of enumerated alternatives to be compared. The most challenging problems in re-architecting in incremental design is right to deal effectively with the combinational explosion.

5.2. Surmounting combinational explosion of re-architecting problem

A genetic algorithm, which is originally founded on Holland (1975), is powerful and broadly applicable stochastic search and optimization technique, and it can be considered as the most widely known types of evolutionary computation methods today (Gen and Cheng 2000). The key idea of genetic algorithms is evolution mechanism of biological organisms in nature according to the principals of natural selection, "survival of the fittest". Individuals that more fit to environment will survive and have a better chance of reproducing. Therefore, the genes from the highly fit individuals become more common in a population, and over time, the population will evolve as composing each individual with well-adapted genes.

In this study, a genetic algorithm is adopted to intelligently search feasible solutions instead of enumerating and comparing all possible alternatives. Although, in principle, the optimal solution to the re-architecting problem can

be found by simple enumeration, in practice it is nearly impossible for realistic systems, where the number of possible candidates of solutions can be extremely high. Therefore, it becomes a major trend to use a genetic algorithm to effectively solve the engineering problems that have a tendency of combinatorial optimization problems (Gen and Cheng 2000). Particularly, several researches utilized the genetic algorithm approach to deal with architecture issues on designing products and processes (Lancaster and Ozbayrak 2007, Meier et al. 2007, Yu et al. 2007a).

There can be several challenges to solve the re-architecting problem with using a genetic algorithm. The first challenge is how to encode a solution of re-architecting in incremental design into a chromosome. The second one is the genetic algorithm for re-architecting should secure feasibility of the generated solution through manipulating a chromosome. Third, fitness for each chromosome should be evaluated on the design engineering contexts. Finally, each step of overall algorithm should be aligned to pursue effectiveness and efficiency on finding an optimal solution. Following sections of this paper focus on how to deal with these challenges.

The present research focuses on developing genetic operators to manipulate the stated difficulties of re-architecting problem. Figure 5-1 shows four core evolutionary operators especially for re-architecting. Encoder create an initial populations of solutions by encoding individual re-architecting strategies, which are solutions of the re-architecting problem, into a string or chromosome on genotype space. Fitness evaluator assesses each individual's fit-

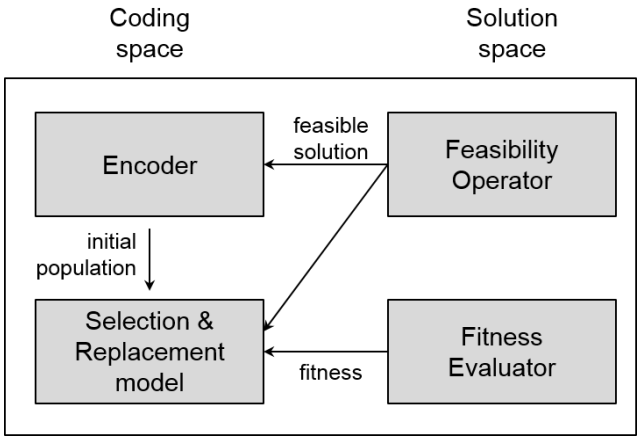


Figure 5-1. Four operators of genetic algorithm for re-architecting

ness with two fitness function, by decoding the genotype solution into phenotype space. Select and replacement model randomly selects individuals and gives opportunity to remain their genes in the population highly fit individuals or solutions, by reproducing offspring. If the generated offspring is more fit than other individual in the population, then it replaces the individual. This evaluation-selection-replacement cycle is repeated until the initial population converges to Pareto-efficient solutions. The Pareto efficient solutions on genotype space is decoded to re-architecting strategies representing new architecture. Meanwhile, when generating initial population and applying genetic operators (cross-over or mutation) to individuals, the proposed algorithm should guarantee feasibility of the resulting solutions. Feasibility operators, on each cycle, identify newly-generated solutions' feasibility and fixes it as feasible. The proposed genetic algorithm for re-architecting is as follows.

5.3. Module-configuration based encoding scheme

How to encode a solution of the particular problem into a chromosome is a key issue when designing a genetic algorithm. Genetic algorithms work on two types of spaces alternatively: coding space and solution space, or in other words genotype space and phenotype space. Therefore, for effective and efficient genetic searching, it is imperative to devise a suitable mapping scheme between solutions in the two different spaces.

Re-architecting strategy for modular systems is mainly composed of two actions: (1) how to compose new modules with carried-over parts (modules) and (2) how to incorporate new components to the new modules. Note that in re-architecting problem, carried-over modules become the element parts composing feasible modules for a new system. This paper encoded a solution by using an J -bit binary string, which has 0-1 integer variables, as the chromosome structure where J is the number of feasible modules composed of carried-over parts. A value of 1 for the jth bit implies that a new component is added (ported) to jth feasible module. In this research, in order to define the feasible modules where new components are ported, *composition matrix* is devised. Particularly, the composition matrix becomes a key to not only encode a solution in genotype space but also decode it to re-architecting strategy in phenotype space.

The composition matrix represents the relationship between composing policy and generated feasible modules from the carried-over parts. Specifically, the composition matrix G of the size $I \times J$ has I feasible composing policies that generate feasible modules from the carried-over parts and J all

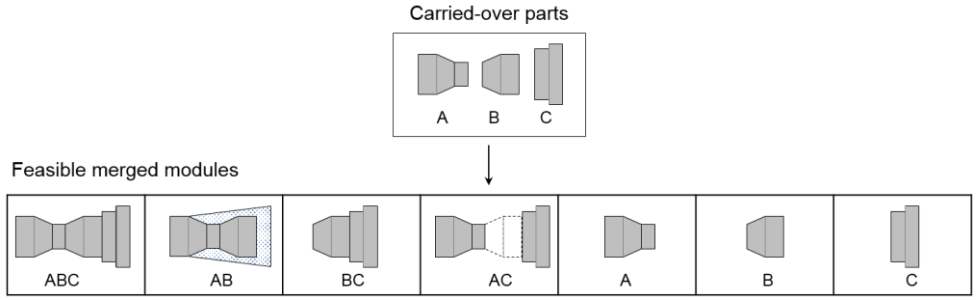


Figure 5-2. Feasible modules from carried-over modules

possible feasible modules. An element G_{ij} equals 1 if policy i generates the feasible module j. Practically, this matrix can be established either by automatically generating algorithm or by designers with addressing the feasibility in merging the carried-over modules and its effectiveness for the new system. Table 5-1 shows an example of the composition matrix for a particular system illustrated in Figure 5-2 that is composed of three carried-over parts, part A, B, and C. In the matrix, the fourth policy, $i = 4$, generates two modules, a feasible module AC of part A and C, and a module B. Meanwhile, not merging any carried-over modules can be one of possible option as shown in policy 5.

Table 5-1. Composition matrix of three carried-over modules

	ABC	AB	BC	AC	A	B	C
policy 1	1	0	0	0	0	0	0
policy 2	0	1	0	0	0	0	1
policy 3	0	0	1	0	1	0	0
policy 4	0	0	0	1	0	1	0
policy 5	0	0	0	0	1	1	1

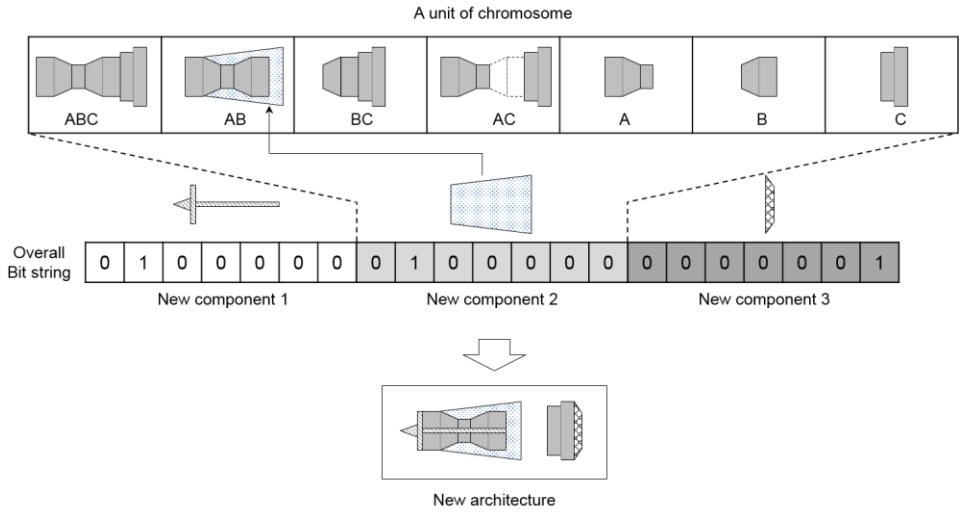


Figure 5-3. Chromosome structure for a re-architecting strategy

Utilizing the composition matrix generated, re-architecting strategy, how to merge carried-over modules and incorporate new components, could be encoded in genotype space. This study uses a $|J|$-bit binary string, as a unit of chromosome structure, for a new component k, where $|J|$ is the number of columns in the composition matrix. A value of 1 for the jth bit implies that the new component k is ported to the jth merged module in the composition matrix. To represent the complete re-architecting strategy, a chromosome should contain information about where all new components are allocated and ported to the merged carried-over modules defined by the composition matrix. Therefore, the author encodes an entire solution through concatenating all of the unit of chromosome structure for each new component; specifically, a solution of re-architecting problem is encoded to $|K|\times|J|$ length of binary

strings, where $|K|$ is the number of new components incorporated to the existing system. Figure 5-3 shows a re-architecting strategy that composes feasible modules AB and C from the carried-over parts (modules), and port new components 1, 2 to the AB module and new component 3 to the C module by using the binary chromosome.

The proposed encoding method holds necessary properties to make a genetic search be effective (Gen and Cheng 2000). First, the mapping between encodings and solutions is one to one; it ensures that no trivial operations will occur when creating offspring. Second, any permutation of an encoding corresponds to a solution. For infeasible solutions generated, feasibility operators, described in following section 5.5, are newly devised to make them feasible solutions. Thereby, any permutation of an encoding can generate a new solution. Third, any solution has a corresponding encoding. This property guarantees that any point in solution space is accessible for a genetic search. Fourth, the meaning of alleles for a gene is not context dependent so that offspring can inherit goodness from parents. On the foundation of the encoding method, fitness evaluator, feasibility operators, and the replacement model are newly proposed in following sections to solve the re-architecting problem.

5.4. DSM utilized architectural fitness

The present paper devises fitness evaluator to evaluate fitness of a solution for unrestrictedly and efficiently crossing between genotype space and phenotype space. A Fitness function of solutions for the re-architecting problem contains

two measures: (1) the induced change efforts on the entire system and, (2) the modularity of the new system. Because required changes according to the chromosome are realized on the existing system's architecture, a key of fitness evaluation is interpreting the chromosome in genotype space to architecting strategy in phenotype space, and thereafter efficiently evaluate the new architecture, which is hypothetically generated by the architecting strategy, respect to two above measures.

The fitness evaluator utilizes design structure matrix (DSM) technique to model modular architecture of the existing system and to analyse its transformations with incorporating new components. DSM highlights the inherent architecture of a design by examining the interactions that exist between its component elements in a square (Steward 1981, Eppinger et al. 1994, Eppinger and Browning 2012). Figure 5-5 represents a modelling example of modular architecture of the existing system and new-component incorporation on DSM. The diagonal matrix represents the element components of the system, and the off-diagonal represents interdependencies, labelled "1" on the cross-sectional cell.

Fitness evaluator calculates the fitness of each gene bit being in the chromosome in advance of evaluating solutions; thereby, when an individual solution has certain chromosome structure, the fitness evaluator identifies which genes in the chromosome are in the solution and then sums fitness of each gene in the solution to compute overall solution fitness. The big picture of operation of fitness evaluator is illustrated in Figure 5-4. To calculate fitness

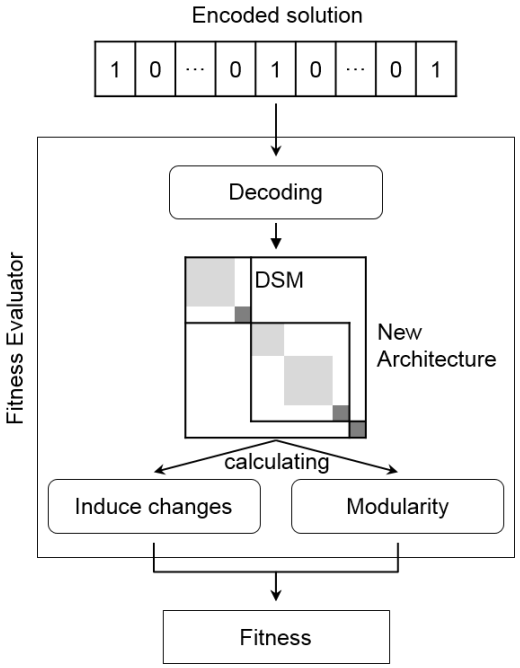


Figure 5-4. DSM-utilized fitness evaluator

of each gene, the fitness evaluator decodes each gene to re-architecting actions; with referring the composition matrix, a gene in the solution can be translated as a set of re-architecting actions of porting a new entering component k into the merged module j that is generated by merging policy i. According to the decoded re-architecting actions, the fitness evaluator hypothetically builds new architecture with utilizing design structure matrix. The induced change efforts and marginal modularity changes incurred by a gene is computed respect to the original existing system. Detailed mechanisms of evaluating each measure are described on design structure matrix as follows.

First, the fitness evaluator computes the induced changes incurred by incorporating a new component into a merged carried-over module. The incremental changes from a new component affect the related components, and interfaces between components propagate changes to the whole system. In theory, it can be assumed that a change occurred in a module does not propagate to other modules to work correctly because the modular architecture include de-coupled interfaces between the modules by standardization (Ulrich 1995). Meanwhile, components in a module include coupled interfaces, and a change triggered by a new component propagates within the module. Therefore, a new component propagates changes to carried-over components because it does not include standardized interfaces; however, the generated changes are kept within the carried-over modules. Figure 5-5 illustrates the mechanism of deriving the induced changes with an example of porting a new component into the selected existing (merged) module. The induced changes in the existing module composed with l components due to new component k takes the following form:

$$\textit{Induced Change}(k \to l \,|\, \text{in merged carried-over module } s) = \sum_{L} \alpha_{k,l} \cdot I_{l,k} \cdot R_l \quad \text{(5-1)}$$

where $I_{l,k}$ is a binary variable, which is marked in the DSM(l, k), that indicates the interdependency between the new component k and the existing component l; the carried-over module g in $G_i = \{g_1, ..., g_n\}$, which is the merged carried-over module set generated by the merging policy i; $\alpha_{k,l}$ is

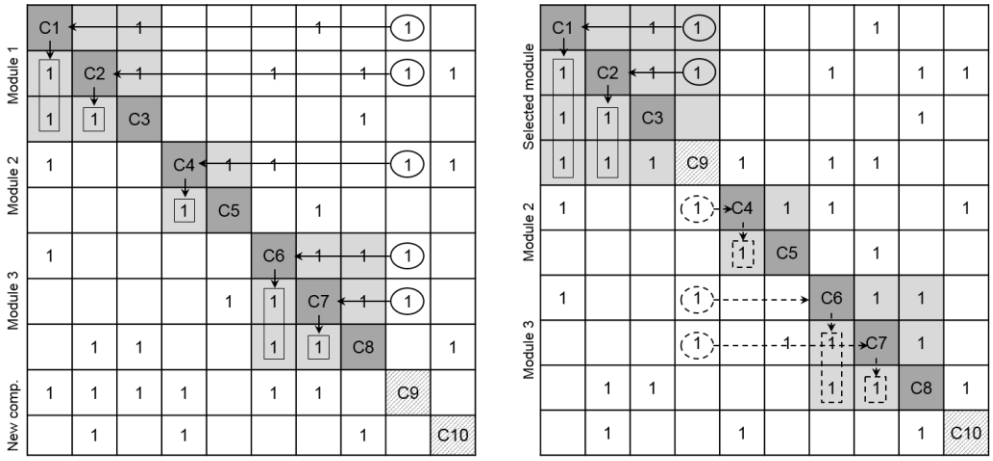


Figure 5-5. Changes due to entering a new component

the change density; and R_l is the number of change propagation routes through the existing component l.

The structure of each modules is characterized by analysing the propagation routes through which a change to any single component causes a direct change to other components in the module. R_l can be represented as

$$Propagation\ \ Routes(l \to m \,|\, \text{in the module}) = \sum_{M} I_{m,l} \tag{5-2}$$

where $I_{m,l}$ is a binary variable that is marked in the $\mathrm{DSM}(m,l)$, that indicates interdependency between change receiver l and its interlinked component m in the module. Note that this research only considers direct change

impacts; nonetheless, it can be expanded by taking into consideration the indirect propagations addressed in other studies (Clarkson et al. 2004, MacCormack et al. 2006).

Porting a new component into the selected modules incurs a cost for standardizing the interfaces related to the surrounding modules. Because the existing modules have already de-coupled the changes from the other components, to enter, a new component should standardize the interfaces that exist outside of the modules to enter. The standardization efforts for the new component's interfaces can be represented as $\sum_{l} t_{k,l} \cdot I_{l,k}$, where $t_{k,l}$ is the coefficient of efforts to standardize the interface for the interdependency $I_{l,k}$ outside of the selected module. Finally, the induced cost to port a new component k into the carried-over module g in $G_i = \{g_1, ..., g_n\}$, which is the merged carried-over module set generated by the merging policy i, is

$$\sum_{G_i} \sum_{L} \alpha_{k,l} \cdot I_{l,k} \cdot R_l - \sum_{G_i \setminus g} \sum_{L} \alpha_{k,l} \cdot I_{l,k} \cdot R_l + \sum_{out\ of\ g} t_{k,l} \cdot I_{l,k} \qquad (5\text{-}3)$$

where l is marked for the existing components that receive changes from the new component k. The first term represents the induced changes on the overall carried-over modules of the existing system, and the second term is the amount of reduction of changes by porting the new component into a selected module g, and the last term computes the standardization efforts for a new component k to enter the selected module g.

Second, fitness evaluator computes marginal modularity changes incurred by a gene respect to the original existing system. In general if modularity is defined as a degree of grouping that puts tightly connected component in the same boundary with no dependencies between these boundaries, a gene, which represents re-architecting actions, alters the grouping of the existing system, and consequently the modularity of the whole system is changed. To measure the modularity of the whole system, the author models a dependency between components l and m similar to MacCormack et al. (2006) as the following forms:

$$Dependency(l \to m \,|\, \text{in same module}) = I_{l,m} \cdot n^{\lambda} \tag{5-4}$$

$$Dependency(l \to m \,|\, \text{not in same module}) = I_{l,m} \cdot N^{\lambda} \tag{5-5}$$

where $I_{l,m}$ is a binary variable indicating the presence of interdependency between components l and m; n is the number of components in the module that l and m are located within; N is the total number of components of the new system that adds new components to the existing system; λ is a setting parameter. These equations model that dependencies in the modules incur a cost lower than those on outside of the modules. The fitness evaluator calculates the marginal modularity changes of new architecture, which is construed by a gene, respect to pre-re-architecting system.

5.5. Infeasible chromosome repair

The important issue on the use of binary strings as the chromosome structure is to guarantee feasibility of the resulting solutions through applying genetic operators (cross-over or mutation) to the binary strings. In re-architecting problem, the individual chromosome should meet two constraints. First, a new component should be ported only to a single merged carried-over module. Second, all new components should be ported to the merged modules made by a single merging policy. To verify the feasibility, the solution matrix S of the size $J \times K$ is built by cutting the binary strings for the individual chromosome in length of J and horizontally concatenating them. Two constraints can be represented as:

$$\sum_{j=1}^{J} s_{jk} = 1, \ \forall k \in K \text{ and } \forall i \in I \qquad (5\text{-}6)$$

$$\sum_{k=1}^{K} \sum_{j=1}^{J} g_{ij} \cdot s_{jk} = K, \text{ for the selected policy } i \in I \qquad (5\text{-}7)$$

where s_{jk} is an element of the solution matrix S ; g_{ij} is an element of the composition matrix G .

5.5.1. Solution feasibility operator

When generating initial population, each individual should satisfy two constraints to ensure feasibility of a solution of the re-architecting problem. The

Table 5-2. Solution feasibility operator

Let

I = the set of the composing policies, $i \in I$

J = the set of the feasible modules, $j \in J$

K = the set of the new components, $k \in K$

S = the set of genes in the solution of an individual string, $s \in S$

M_i = the set of feasible modules generated by the i th policy; the columns in the solution of i th row in the composition matrix, $m_i \in M_i$

w_i = the number of feasible genes in the solution respect to the policy i; the number of new components that are ported to the i th row-feasible modules.

A_i = the set of feasible genes in solution respect to the policy i, $a_i \in A_i$

P = the entire population composed of individual p, $p \in P$

For the individual p that presents its maximum w_i is less than $|K|$

i. Randomly select a composition policy $i \in I$.

ii. Identify the set of infeasible genes by $F_i := S - A_i$, and set $S := S - F_i$.

iii. For each gene of $f_i \in F_i$

(a) Randomly select the feasible module of $m_i \in M_i$.

(b) Add $(k-1) \cdot J + m_i$ to *S*, where *k* satisfies $(k-1) \cdot J + 1 \le f_i < k \cdot J$.

iv. S is now a feasible solution that satisfies two mentioned constraints.

encoder generates the initial population by randomly allocating a "1" to a unit of chromosome structure, which has a $|J|$-bit binary string, as represented in Figure 5-3; however, because the gene in solution is selected randomly, many of the generated individuals should be supposed to violate the two feasible conditions above mentioned. Therefore, the author proposes a heuristic feasibility operator that identifies infeasible individuals and fixes them to make the entire population feasible. The algorithm of the operator is shown in Table 5-2. Step (i) randomly selects the composing policy that generates the merged module where the new components are ported. Step (ii) identifies the infeasible genes to be fixed, and step (iii) makes each gene feasible by randomly selecting a feasible module.

5.5.2. Re-architecting feasibility operator

In this study, to crossover the genetic information, the author uses the fusion operator (Beasley and Chu 1996) that produces just a single child. The fusion crossover operator works by combining genes of the two parent strings to produce a single child strings. When combining two parent strings, the choice of whose gene values are passed to the child should be made based on the relative fitnesses of the two parents. Specifically, it becomes more likely for the gene of the highly fit solution to be inherited to the next generation. Meanwhile,

mutation usually operates by inverting each bit of individuals with a small probability. This paper uses a mutation rate of $1/n$, which is suggested as a lower bound on the optimal mutation rate (Bäck 1993), where n is the length

Table 5-3. Re-architecting feasibility operator

ns_k = the number of genes in the solution on a unit of chromosome structure, which is a $|J|$-bit binary string between $(k-1)\cdot|J|$ and $k\cdot|J|$ of the individual chromosome; i.e., the number of 1-value bits that a new component k has

nc = the number of new components that satisfy $ns_k \geq 1$

i. For the case of $nc := |K|$

If $ns_k = 1, \forall k$

Obtain S through the solution feasibility operator

Else if for k, $ns_k > 1$

(a) find M_i that has maximum w_i

(b) select m_i in M_i that minimizes change efforts/dependencies

(c) add $(k-1)\cdot|J| + m_i$ to S

ii. For the case of $0 < nc < |K|$

For k, $ns_k > 1$ or $ns_k = 0$

(a) find M_i that has maximum w_i

(b) select m_i in M_i that minimizes change efforts/dependencies

(c) add $(k-1)\cdot|J| + m_i$ to S

iii. For the case of $nc := 0$

Generate a new feasible solution

* The other variables use same definitions represented in Table 5-2.

of the chromosome.

A child strings generated by crossover and mutation operators might violate the feasibility constraints of Eq. (5-6) and (5-7). To make a child string feasible, here the author proposes a heuristic operator called the *re-architecting feasibility* operator that not only fixes the infeasibility of the solution but also provides a local optimization step in an attempt to make the genetic algorithm more effective. Note that the re-architecting feasibility operator is different to the solution feasibility operator above in that it should deal with two constraints of the solution, whereas the solution feasibility operator deals with the second constraint.

The steps required to make each child feasible involve the identification of all genes that violate two constraints of the re-architecting problem in a solution and the fixation of the infeasible genes to satisfy the two constraints. Once infeasible genes are fixed and a solution becomes feasible, several possible fixation alternatives can exist. Therefore, a local optimization step is applied to identify the best fixation alternative by comparing their fitnesses. Also, it can effectively exploit the solution space. The detailed algorithm is shown in Table 5-3. The algorithm firstly classifies an infeasible individual as one of type (i), (ii), and (iii) according to chromosome structure. Then, the algorithm identifies the infeasible units of chromosome structure, and it fixes them to be feasible with considering fitnesses of possible fix alternatives.

5.6. Bidirectional evolutionary algorithm

The re-architecting in incremental design is the multi-objective problem that deals with more than one criterion simultaneously; the two criteria are (1) minimizing the induced change efforts on the entire system and (2) maximizing the modularity of the entire system. Because the two criteria are incommensurable and conflict each other, there does not necessarily exist a single best solution respect to the both objectives. A certain solution is best in minimizing the induced change but can be worst in maximizing the modularity. Therefore, for such solutions the proposed genetic algorithm identifies *non-dominated solutions* or *Pareto optimal solutions,* which cannot achieve improvement in any objective function without sacrificing the other objectives (Gen and Cheng 2000).

The present paper devised a bidirectional evolutionary algorithm to accommodate incommensurable multiple objectives with an inspiration from the vector-evaluated genetic algorithm (Schaffer 1985). Figure 5-6 illustrates the schema of bidirectional evolutionary algorithm devised for the re-architecting problem work in each generation. The each loop of the genetic algorithm is composed of four major steps. The first step is dividing (halving) overall population into two subpopulation to apply the genetic operators to individuals according to two objectives. Therefore, one subpopulation is used for finding the solutions that minimize the induced change efforts, and the other subpopulation is for identifying the solutions that maximize the modularity of the new system. The second step is selecting two individuals and reproducing the child in each subpopulation. Third, a new feasible child solution replaces a randomly chosen member (one with an under-average fitness value) in the each

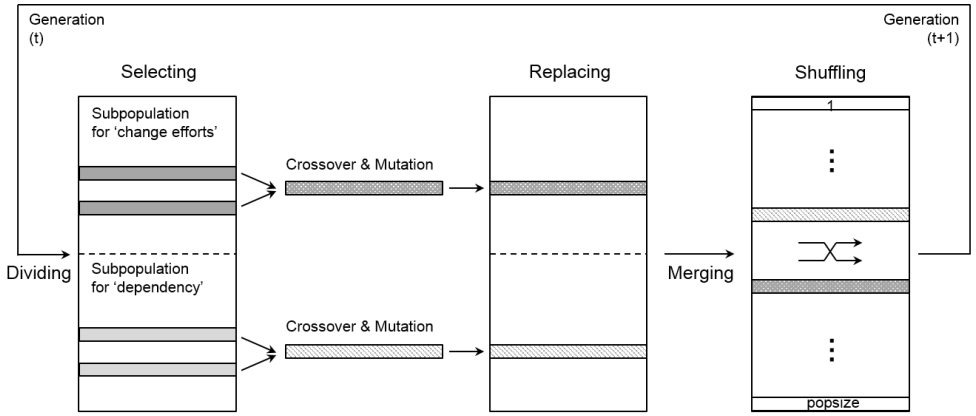


Figure 5-6. Bidirectional evolutionary algorithm

subpopulation. The final step is merging the dichotomous subpopulations into a single population and shuffling all individuals in the unified population to prevent solutions to evolve in one direction. A loop composed of four steps should be repeated until all individual solutions in the entire population are converged to Pareto efficient solutions.

Parent selection is to assign reproduction opportunities to each individual in the subpopulation. If an individual has high fitness, the individual would get more opportunities than others. The author adopted the binary tournament selection method that works by forming two pools of individuals, each consisting of two individuals drawn from the subpopulation randomly. The proposed genetic algorithm forms two pools for each subpopulation as represented in Figure 5-6. Two individuals with the best fitness, each taken from one of the two tournament pools, are chosen for mating. Crossover operator, particularly utilizing the fusion operator, produces a single child unlike

the one-point and two-point crossover operators where two children are produced. Mutation is applied to each child after crossover. The above selection procedures are conducted in each subpopulation where pursuits two different objectives in re-architecting; one subpopulation uses the induced change efforts as the fitness function, and the other one uses the modularity as the fitness function. Therefore, the each child produced from the two different subpopulations may evolve in different directions. This can make solutions simultaneously pursuit two criteria.

Table 5-4. Replacement of dominated population

Let
P_r^l = the subpopulation r on lth generation , $P^l = \{p_1^l, p_2^l\}$
r = the number of subpopulation, r=[1,2]
C = the induced change efforts
D = the dependency (the measure of the modularity)
l = the number of the generation
Z_r^l = the set of newly reproduced children from each subpopulation r on lth generation, $z_r^l \in Z_r^l,\ r = 1, 2$
$Z_r^{0,l}$ = the set of the randomly chosen individuals from each subpopulation r, $z_r^{0,l} \in p_r^l$

(i) For each generation l
  If $C(z_r^l) < C(z_r^{0,l})$ and $D(z_r^l) < D(z_r^{0,l})$
    Replace $z_r^{0,l}$ with z_r^l in p_r^l.
    Set $p_r^l = p_r^l + z_r^l - z_r^{0,l}$.
  Else
    Set $p_r^l = p_r^l$.
(ii) Merge p_1^l and p_2^l, shuffle it, and set $l := l + 1$.

Once new feasible children have been generated, each child will replace a randomly chosen individual in the each subpopulation. Our algorithm randomly chooses an individual in the subpopulation and compares the fitnesses of the produced child and the chosen individual. If the child dominates the chosen individual in the criterion space composed with the induced change effort and the modularity, the child will replace the chosen individual. The algorithm is simply represented as Table 5-4. Note that the algorithm makes an individual be replaced only if the solution is dominated in terms of two criteria of the re-architecting to improve the solutions without sacrificing any one of two objectives. With this algorithm, the children evolved in both objectives will replace the existing solutions in the population, and the overall population can improve with simultaneous pursuit of two objectives. The algorithm terminates when there is no more Pareto improvement in the population.

5.7. Application to hydrogen-fueled internal combustion engine

The algorithm presented in this paper is coded in MATLAB® and demonstrated with hydrogen-fueled internal combustion engines (H_2ICE), which are developed based on the traditional internal combustion engines (ICEs), as a subject for incremental design.

To examine the proposed genetic algorithm, the author represents the architecture of H_2ICE system in phenotype space with using design structure matrix or DSM technique. Figure 5-7 presents elements that compose H_2ICE

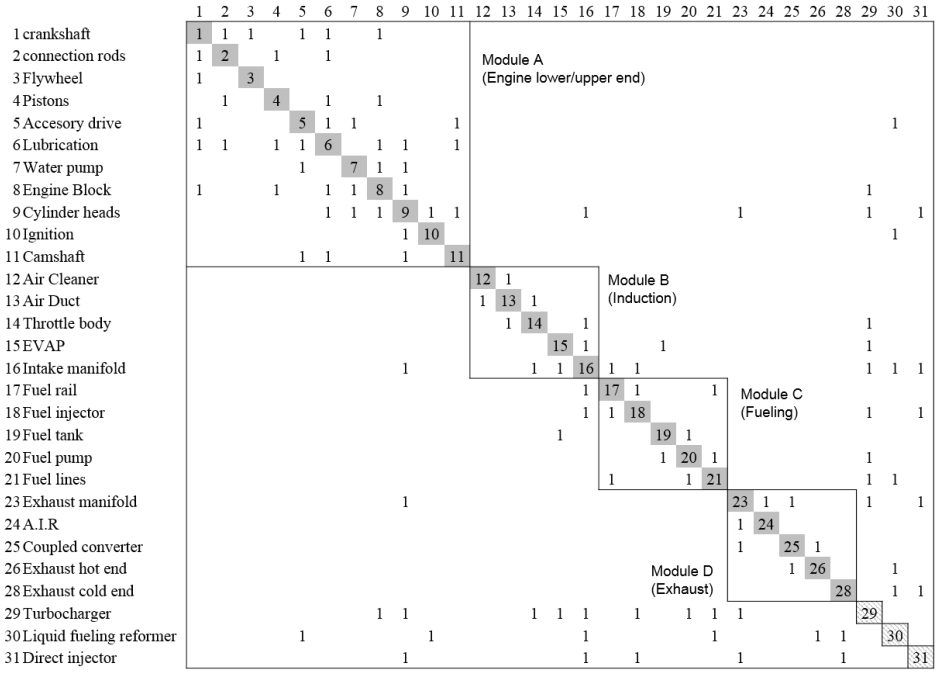


Figure 5-7. DSM of Hydrogen-fuelled internal combustion engines

system and their relations on a DSM. Note again that the diagonal matrix represents the element components of the system, and the off-diagonal matrix represents the relations between the components with a binary variable $I_{a,b}$; if component a and b are related each other, $I_{a,b}$ becomes "1". A block that binds the several components into a group represents a module that carry out an independent function. In the figure, four modules compose the conventional ICEs: module A is lower/upper engine; module B is the induction system; module C is the fuelling system; and module D is the exhaust system. To meet requirement of H_2ICEs, three new components, component 29 to 31 il-

lustrated with a deviant crease line, are added to the existing ICEs; new technologies that focus on power density, NO_x emissions, and thermal efficiency. The relations between new components and the existing modules are postulated based on an advanced reciprocating engine concept that encompasses the several advanced technology options in the near term (Peschka et al. 1992, Jaura et al. 2004, Rottengruber et al. 2004, White et al. 2006). Note that the existing engine system composed of component 1 to component 28 has well defined modules in that there are very few dependencies between the existing modules, as evidenced by the fact that much of the DSM out of the modules is "white space". However, as the new components are added to the existing system, the entire system's efficient architecture is broken and becomes inefficient. Therefore, to establish the efficient new architecture for the H_2ICEs, re-architecting the existing system is necessarily required.

To solve the re-architecting problem for H_2ICEs with a genetic algorithm, the first step is to encode the re-architecting solutions implying how to configure the carried-over modules from the baseline system and the new components added to meet changed requirements in the phenotype space to the genotype space. The composition matrix is established as Table 5-2 with five given alternatives of the feasible merging policies. Columns of the composition matrix represent all possible merged modules that can be generated from the carried-over modules. In this case study, the conventional ICEs have four major modules (parts), A, B, C, and D modules illustrated in Figure 5-7, and overall 15 feasible modules are possibly generated. Meanwhile, rows of the

matrix represent the composition policies that generate feasible modules with the carried-over modules.

Based on the composition matrix, the proposed genetic algorithm encodes a solution of the re-architecting problem in genotype space and generates the initial population. Each of three new component added to the conventional ICEs has a 15-bit binary string as the chromosome structure, where a j th bit is in solution if the new component is ported to the j th merged module in the composition matrix. Because there are three new components, the entire solution is encoded to a 45-bit string. The proposed algorithm generates initial population with retaining feasibility of the solutions through the solution feasibility operator, presented in section 5.5.1. The population size is set to 100, when the author considers the number of all possible alternative architectures to be compared is $\sum_{i}^{I} n(i)^{|K|}$, where K is the set of the new components; $n(i)$ is the number of merged modules generated with the merging policy i, and it is sufficient to cover the entire solution domain.

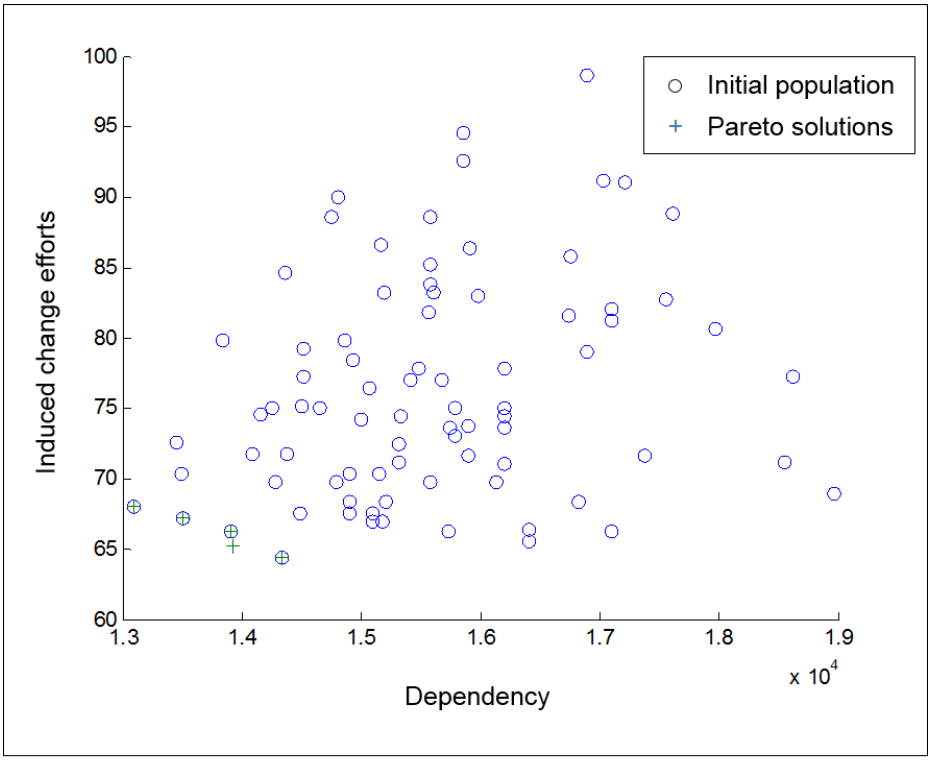
In each generation the bidirectional evolutionary algorithm applies a series of genetic operators to the population as illustrated in Figure 5-6. The overall population is divided into two subpopulations to reproduce solutions pursuing each objective of two. In each subpopulation, two individuals are selected by identifying higher fit one on the binary tournament and produce a child by applying the fusion cross-over operator to two parent solutions. Thereafter, the child's chromosome is inverted with a small mutation rate of 1/45, where 45 is the string length of the chromosome. In each subpopulation,

Table 5-5. The composition matrix for H_2ICEs

	ABCD	BCD	ACD	ABD	ABC	CD	BD	BC	AD	AC	AB	D	C	B	A
policy 1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
policy 2	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
policy 3	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0
policy 4	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0
policy 5	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1
policy 6	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0

Table 5-6. Solutions and their fitnesses

New components	Turbocharger															Liquid fueling reformer														Direct injector														Fitness	
Merged modules	ABCD	BCD	ACD	ABD	ABC	CD	BD	BC	AD	AC	AB	D	C	B	A	ABCD	BCD	ACD	ABD	ABC	CD	BD	BC	AD	AC	AB	D	C	B	A	ABCD	BCD	ACD	ABD	ABC	CD	BD	BC	AD	AC	AB	D	C	B	A	Change	Dependency
Solution 1	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	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	66.2	13900.6
Solution 2	0	0	0	0	0	0	1	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	0	0	0	0	0	0	1	0	0	64.4	14327.5
Solution 3	0	0	0	0	0	0	1	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	0	1	0	0	0	0	0	0	0	65.2	13916.4
Solution 4	0	0	0	0	0	0	1	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	0	0	0	0	0	0	0	1	0	0	67.2	13505.2
Solution 5	0	0	0	0	0	0	1	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	0	0	1	0	0	0	0	0	0	0	68	13094.1



Fitness results we report use $\alpha = 2$, $t = 1.3$, $\lambda = 1.6$. It is necessary for the analyzer to decide these variables to some value while considering the characteristics of the object system and the sensitivity of the value

Figure 5-8. Pareto efficient solutions

the child replaces the individual, randomly chosen, if dominating it in regard to two criteria. Finally, two subpopulations are merged and shuffled to prevent solutions to evolve in one direction. By repeating the loop through 1500 generations, the re-architecting genetic algorithm obtains Pareto optimal solutions of the re-architecting strategy.

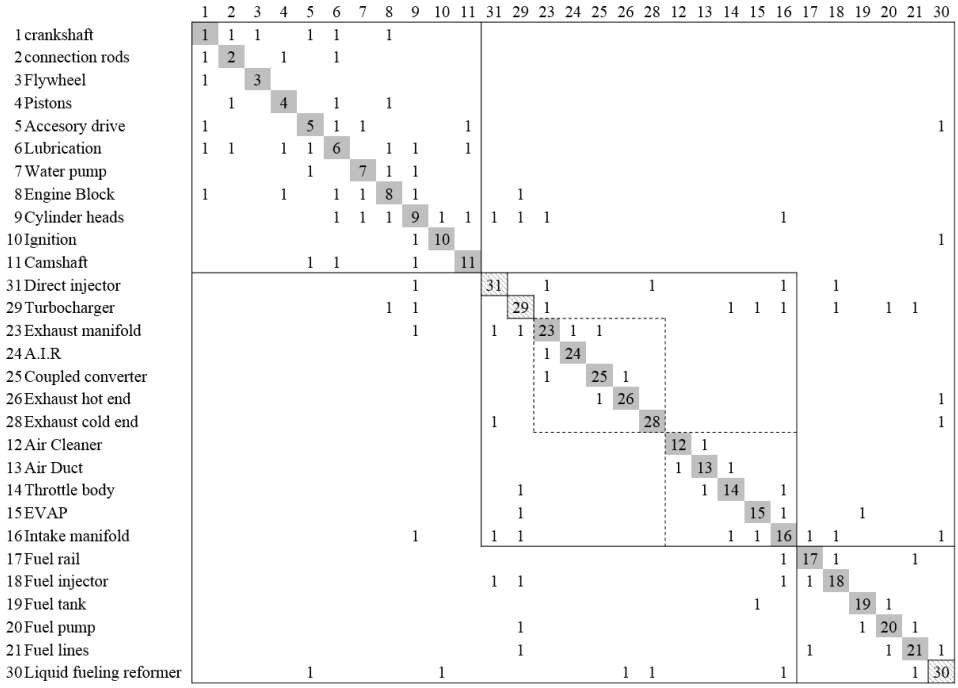


Figure 5-9. New architecture for H_2ICEs (solution 3)

The re-architecting genetic algorithm identifies five non-dominated solutions. Figure 5-8 plots the initial population and the Pareto optimal solutions with two fitness value. The x-axis represents the dependency for the individual solution, and the y-axis represents the induced change efforts. The circle is marked for the initi al population, and the cross marks Pareto optimal solutions that are converged to only five points throughout 1000 generations. Because the re-architecting modular systems in incremental design pursues minimal changes and minimal dependency on the entire system, the Pareto optimal solutions are converged to the left low part of the Figure 5-8. Each point for Pareto solutions cannot improve two fitness simultaneously without sacrificing any one of them, and it means that the convergence is completed.

The proposed genetic algorithm makes a good balance between two important issues with respect to search strategies: exploiting the best solution and exploring the search space (Booker 1987). Four over five of the converged solutions are overlapped by the initial individuals (marked with circles); meanwhile, a single converged point is emerged from blank. It means that even though the initial population does not fully cover all possible solutions of the problem, the proposed algorithm can search and identify the non-dominated solutions through utilizing evolutionarily mechanism such as crossover, mutation, selection, and replacement. Moreover, when the size of the problem is too big to sufficiently cover all possible solution space, the proposed evolutionarily algorithm can provide the chances to exploit best solutions from the empty space.

The Pareto optimal solutions in genotype space can be decoded to re-architecting strategies in phenotype space with using the composition matrix. The five Pareto optimal solutions and their fitness are represented in Table 5-6; each solution chromosome is composed of 45-bit strings that encode the information of where three new components are ported to merged modules, generated by the merging policies. To easily decode and understand the genotype optimal solutions into phenotype solutions, the author presents the names of the new components and merged modules on the top of the Table. Note that the merged modules written in the Table come from the composition matrix of Table 5-5 in same order. Solution 1 is decoded as merging the carried-over modules of the conventional ICEs with the merging policy 1, which generates A, B, C, and D modules; porting turbocharger to D module; porting liquid

fueling reformer to B module; and porting direct injector to C module. Solution 3 is decoded as merging the carried-over modules by the merging policy 5, which generates BD, C, and A modules; and then porting turbocharger and direct injector to BD merged module; porting liquid fueling reformer to C module. Figure 5-9 represents new architecture for H_2ICEs according to the solution 3 on design structure matrix. The rest of the genotype solutions can be decoded re-architecting strategies, and the new architectures corresponding to each solution can be built on design structure matrix, including composition of modules and interdependencies among them.

With the proposed bidirectional evolutionary algorithm, two criteria of the re-architecting in innovative redesign can be simultaneously pursued; the two criteria are (1) minimizing the induced change efforts on the entire system and (2) maximizing the modularity of the entire system. Because the two criteria are incommensurable and conflict each other, there does not necessarily exist a single best solution respect to the both objectives. The enumeration method utilized in Chapter 4 might be a kind of sequential optimization method. The algorithm generated alternative architectures that can incorporate new components (technologies) with minimum induced change efforts. Thereafter, the most modular alternative is chosen among them. However, a solution architecture may be best in modularity but worst in induced change efforts. Therefore, there usually exist a set of solutions for the multiple-objective case which cannot simply be compared with each other. The bidirectional evolutionary algorithm proposed in Chapter 5 does not sacrifice solutions with respect to a single criteria. Rather, it tried to identify Pareto optimal solutions,

no improvement in any objective function is possible without sacrificing at least another objective function. Indeed, with comparing solutions identified by the algorithms respectively proposed in Chapter 4 and 5, the Pareto optimal solutions from Chapter 5 subsume the optimal solution of Chapter 4 (they were identified in the same parameter setting; α=2, t=1.3, and λ=1.6). The optimal solution obtained from Chapter 4 corresponds to the solution 5 of Table 5-6 (also represented in Figure 5-8), which has minimum dependency among the five Pareto optimal solutions of Chapter 5. However, the other solution 1 to 4 are better than the solution 5 in respected to the induced change criteria.

Naturally, the bidirectional evolutionary algorithm can be easily extended to select one of the nondominated solutions as a final solution to re-architecting modular systems in innovative redesign. Conceptually, the preference intends to give an order to the incomparable solutions within the Pareto efficient solution set by using designers' value judgments on objectives. The preference reflects either designer's trade-offs among objectives or an emphasis on particular criterion according to prior experiences of the architecting problems. With a given preference the bidirectional evolutionary algorithm can be modified to order the alternative solutions in the nondominated set, and then the algorithm can obtain a final solution, which is the usual outcome of a decision-making process. Note that an optimal solution obtained in Chapter 4 might be a final solution with extreme preference on the modularity.

5.8. Summary

The family of combinatorial optimization problems is characterized by enumerating a finite number of feasible solutions. In principle, identifying the best solution among the finite alternatives could be done by simple enumeration. However, real world problems are much more complicated, and the enumeration is frequently an impossible approach to conduct because the number of feasible solutions generated by combination explodes. Developing a modular product can be classified as a combinatorial optimization problem as each company works endlessly for the optimality of their products (Kamrani and Gonzalez 2003, Yu et al. 2003, Yu et al. 2007b). Re-architecting modular system is the strategy-establishment problem how to re-arrange carried-over modules with decision of whether they embrace the new components in their module boundary or not. The exact algorithm to find an optimal solution of the re-architecting problem is simply enumerating all feasible architecture alternative, which are configured with carried-over modules and new components. Meanwhile, heuristic methods can search only a portion of the solution space and find good solutions to a problem based on a number of constraints. These heuristic techniques are more efficient, as they give solutions faster than the enumeration methods discussed before. However, the trade-off is arisen and consists in giving up the possibility of getting an exact optimal solution in order to achieve acceptable results within a reasonable amount of time. Alternatively, genetic algorithms can search the solution space more broadly, giving the user a better chance of getting an optimal solution with less effort than the enumeration methods and heuristic methods (Gen and Cheng 2000). In

addition, with forcing solutions to simultaneously evolve in multiple directions, the genetic algorithm can be conveniently utilized to pursue two objectives, which are minimizing the induce changes and maximizing modularity of the entire system, in re-architecting modular system in innovative redesign.

To break through the combinational explosion, a genetic algorithm is adapted to efficiently identify solutions of the re-architecting problem without searching whole of the solution space. The proposed genetic algorithm encodes re-architecting strategies representing where new components is ported to the existing module that is generated by how to merge the carried-over modules with using binary bit stings. Also, the fitness evaluator is devised to calculated fitness of a solution with unrestrictedly crossing between genotype space and phenotype space. To deal with breaking feasibility arisen through applying the genetic operators such as crossover and mutation, the proposed genetic algorithm devises the heuristic-based feasibility operators. Moreover, bidirectional evolutionary algorithm is proposed to pursue two criteria simultaneously, and Pareto optimal solutions can be identified through several generations.

Chapter 6. Conclusions and Future Works

This study developed methodologies for establishing design strategies in incremental design based on the consideration of the product architecture. These design strategies determine a consistent set of design targets, what to achieve through a new product, and implement them efficiently in physical domain. This is, overall design information playing in different design domains are streamlined and integrated on the product's architecture. Especially, in incremental design, as the existing product is the foundation of the new product, it is imperative to effectively and efficiently meet new requirements within the carried-over design space. On the functional domain, the design space can be defined of specifications (functional elements) of the previous products. Meanwhile, on the physical domain, the modular architecture or the component arrangement of the existing system become the design constraints for a new system. Therefore, this paper understood the existing product architecture and established design strategies for the different design domains.

While traditional design research has concentrated on creativity in the early phase of design and on creativity in very open-ended design tasks, however, in practice many design projects concern the modification or incremental design of existing systems to meet new needs and restrictions. Therefore, the existing system acts as design constraints in incrementally designing new sys-

tem. In order to inherit competitiveness of the previous products, the new system carry over the core parts of the existing system. As a necessity, the characteristics from the carried-over parts should become a kind of constrains to design new system. When considering the domains where design tasks are conducted, the constraints can be described in different forms: on the functional domain, the interrelationships between specifications define the design space characterizing the existing products; on the physical domain, the arrangement of the physical components and the interdependencies between them act as the constraints in architecting new systems with incorporating new components.

In incremental design, one of the most important design tasks is the determination of design targets that precisely describe what the new product to do. Particularly, as the products get more complex, it becomes harder to understand the existing system in terms of interrelationships between functional elements. Moreover, new design targets should be identified to maximally value the customers within the feasible design space. To overcome the difficulty in understating the existing system, this paper proposed the data-driven methodology to determine a consistent set of design targets of the complex systems through a new vehicle-planning case.

It is important to design new-car models that are both sustainable in uncertain markets and technically feasible. For that, this paper proposed a methodology that determines vehicle-level specifications by balancing market environments and engineering feasibility in the early stages of the vehicle development processes using a network of empirical models. The proposed

methodology establishes engineering design constraints that define the strategic feasible space within which next-generation automobile specifications must be formulated. Moreover, the customer utility derived from a new vehicle concept is built as a function of preferences for each attribute so as to enable the design of a new-car model that sells well regardless of market-environmental changes.

The methodology, though general in nature, was applied specifically to preliminary-stage vehicle design processes. From the research perspective, the proposed methodology presents a new, historical-data-based statistical approach that effectively surmounts the difficulty of mechanically understanding complex systems. In the industrial realm, this methodology will enable designers to plan complex products for new concepts based on the quantification of information rather than the intuition of experts. The proposed model, by taking an engineering perspective on the constraints imposed by the interrelations among specifications and technology advancement, can provide feasible prediction values. Moreover, through quantification of the target market's customer preferences as well as an understanding of how attribute preferences shift according to market-environmental changes, very objective and reasonable determinations can be obtained.

There are several directions for future work. One avenue is to enhance forecasting ability for the next-generation vehicle. The proposed model tried to extrapolate technology advancement path and to predict customer preference changes on particular attributes of vehicles with utilizing external data. Investigating new relative data sources that can extend or improve prediction

power of DOVES would be a very interesting and topical research. Another topic of interest is to develop the methods of collecting external data. Indeed, some researches have attempted to automatically gather a large data set (Lee and Bradlow 2007, Giffin et al. 2009, Gokpinar et al. 2010). More research would need to be done to elicit essential data from flood of information to improve utilization of information and operation efficiency.

The other important issue in new product design is how to effectively and efficiently implement the design targets on the physical domain. Especially in incremental design, not only the core components but also their arrangement are carried over to the new system. Meanwhile, new requirements for the product cascade down to the physical domain through specifying and realizing with physical components. As a necessity, the stabilized architecture of the existing system is affected, and the design changes are required. Therefore, it is imperative to re-arrange the entire system for efficiently incorporating new components while retaining carried-over parts.

In this paper, a new methodology for re-architecting an existing system through accommodating newly required changes is introduced; additionally, the methodology is demonstrated using the example of hydrogen-fueled internal combustion engines. If the existing system is comprised of well-defined modules, incremental design should not only change it as little as possible but should also retain the modular architecture. The proposed methodology defines re-architecting operators with DSM to systemically address the modules in the process of architecting. Each operator, as applied to the existing modules, generates a large number of alternatives for the new architecture. To determine

the optimal architecture among those alternative architectures, the proposed methodology provides algorithms for the operator's activation. These algorithms are formulated with Matlab® and applied to a new engine concept wherein three new technologies were incorporated into traditional internal combustion engines. The results of this case study satisfactorily demonstrated the necessity of re-architecting in incremental design, even though the existing system had a well-defined modular architecture.

The uniqueness of the present study is the consideration of the induced changes and their propagation in modular design strategy. As the flexibility of the modular system, design methodology of how to reconfigure the existing modules should be necessary to effectively and efficiently respond to changed requirements. The bulk of modular design methods have focused on identification of a single, final modular architecture with considering dependencies and similarity (compatibility) between modules (Gershenson et al. 2004, Salvador 2007, Campagnolo and Camuffo 2010). However, the main problem in incremental design of modular system is *change*. Understanding of what changes are induced in the existing system due to new components and how the induced changes propagate to the entire system is necessary to establish new modular architecture. This paper modelled the induced changes and their propagations, and proposed new modular architecture that induces minimum change effort to the existing system. This approach focuses on the transformation itself from the existing system to the final result rather than dramatically identifies a single, final result.

Many studies, acknowledging the importance of incremental design, have concentrated on assessing the impacts of new-technology incorporation into existing systems. Investigations specifically concerning incremental-design methodology, contrastingly, have been relatively scare. In the academic perspective, our methodology clarifies the architectural changes that are incurred in incremental design and provides a solution to the problem of re-architecting an existing system to accommodate newly required changes. In practice, designers can refer to the core algorithm of the module operation to establish their own methodology, while considering the characteristics of the firm and product. Moreover, the best re-architecting strategy could be determined depending on how the design philosophy weighs the importance of changing the existing system as little as possible versus making the new system a highly modular product.

In principle, the optimal solution to the re-architecting problem can be found by even enumeration or simple heuristics; however, in practice it is frequently impossible where the number of feasible solutions can be extremely high. Re-architecting modular systems in incremental design has such difficulties; especially for practical problem of realistic size, the explosive increase of feasible alternatives compared becomes the most challenge. In detail, as the modular systems get bigger, the number of generated feasible modules brings out the exponential increase of the number of enumerated alternatives to be compared. The most challenging problems in re-architecting in incremental design is right to deal effectively with the combinational explosion.

To surmount the difficulty arisen in the proposed methodology, the present paper proposes the genetic algorithm for re-architecting modular systems in incremental design to overcome the challenges arisen by getting the systems bigger in practical cases. Re-architecting methodology proposed by Kang and Hong (2013) compares all possible alternative architectures come from a nature of the modular systems for the new system. In practice, as the modular systems get bigger, the exponential increase of the number of alternative to be compared becomes the most challenge on solving the re-architecting problem. Therefore, to break through the combinational explosion, a genetic algorithm is adapted to efficiently identify solutions of the re-architecting problem without searching whole of the solution space. The proposed genetic algorithm encodes re-architecting strategies representing where new components is ported to the existing module that is generated by how to merge the carried-over modules with using binary bit stings. Also, the fitness evaluator is devised to calculated fitness of a solution with unrestrictedly crossing between genotype space and phenotype space. To deal with breaking feasibility arisen through applying the genetic operators such as crossover and mutation, the proposed genetic algorithm devises the heuristic-based feasibility operators. Moreover, bidirectional evolutionary algorithm is proposed to pursue two criteria simultaneously, and Pareto optimal solutions can be identified through several generations.

This study contributes to the literature on architecting a product in general and on the specified methodology for conducting re-architecting in incremental design in particular. Whilst many studies have noted the importance of

incremental design and focused on assessing impacts on the existing system, the proposed methodology provides architecting strategies to deal with incremental design in active manners. Particularly, to efficiently solve re-architecting problem of complex system in real world, genetic algorithm is adapted through devising new operators specialized for the problem. Because real systems are composed of a number of modules, it is indeed necessary to efficiently identify the optimal architecture for new system without exploring all feasible architecture solutions. Finally, this research could provide an evidence to assure that genetic algorithm is effective and efficient to solve optimization problems from engineering designs that are very complex in nature and quite difficult to solve by conventional optimization techniques.

The developed re-architecting methodology opens up a number of avenues for future study. With respect to methods, I believe that the DSM technique provides a powerful lens through which to operate modules and components in complex systems. Although my focus was the question of whether interdependency among components exists or not, the proposed methodology can be generalized by addressing the types of interactions among components in four dimensions and quantifying them on a 5-point scale (Pimmler and Eppinger 1994). With respect to re-architecting strategy selection, the methodology searches for the best alternative by iteratively comparing all possibilities. Therefore, there is value in establishing heuristic rules for the iterative process if differences in adopting our methodology to several types of products can be

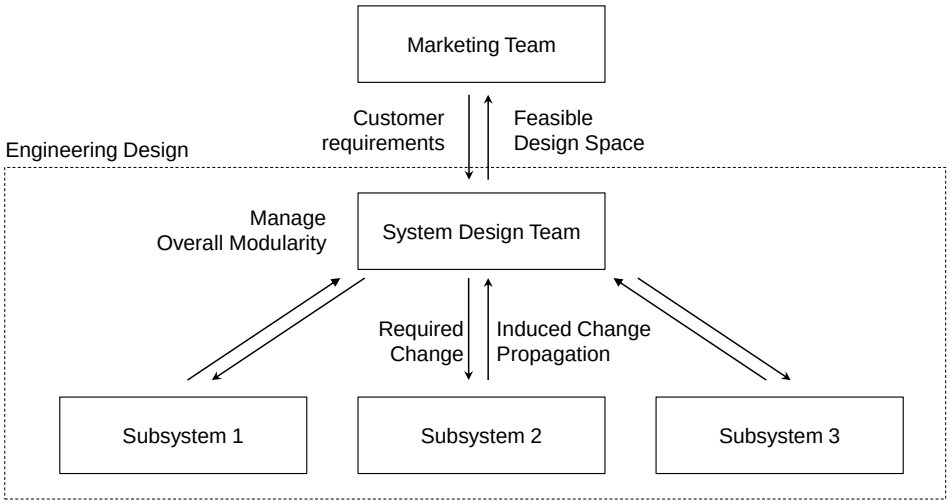


Figure 6-1. Cross-functional product development methodology

identified. Finally, using realistic dependency and cost data, this research can be extended to many practical cases that have become hot issues.

In practice, the presented design methodologies in this thesis can be integrated and utilized for cross-functional product development. Figure 6-1 shows organizational hierarchy according to the allocated design tasks. Mainly, marketing team and engineering design team interact each other to make a plan for a new product system. Particularly, a consistent set of design targets should be identified through organically exchanging the design information between two parts. Marketing team understands customer requirements, related regulations, and environmental factor and suggests design targets according to the understandings. The engineering design teams verify whether the suggested design targets are defined in feasible design space or not. If it is infeasible, the engineering team demand marketing team to modify the design targets to be in feasible design space. Determination of the design targets is

conducted with the adjustment of design information from each side in iterative manner. The present thesis proposed the determination methodology that mimics and models the realistic design process to identify a consistent set of design targets, which both marketing team and engineering design team agree. This can help product planning, which has been conducted based on the designers' partial experiences, be accomplished in scientific manner.

In engineering design team, it is imperative to effectively and efficiently implement the identified design targets to the physical system. System design team cascades down the required design targets to the subsystem design team. Particularly, most of new product development is on the foundation of the existing system. Changed requirements should not only generate design changes to the existing system but also break the stabled modularity. Therefore, on the system level, system design team manages overall modularity and induced changed. The present thesis proposed the re-architecting methodology for modular system in innovative redesign. With the methodology, engineering design team can identify the implementation strategy of the existing system to meet new requirements in systemic manner.

The overarching message of this paper is that strategic product development is linked to the architecture of the product, so designer should address the architecture to effectively and efficiently accomplish the design goals. Particularly, as considering that new product is developed on the foundation of the existing product in many cases, it is imperative to comprehensively understand the architecture of the existing system. According to design stages on

new product development, the architecture is utilized in different ways to establish design strategies. On very early stage of development process, the functional arrangement of the architecture is to be reference to plan a new product. Meanwhile, arrangement of physical components and their interfaces of the existing system should be referred for new system to minimize change efforts arisen by transformation. Listed here are questions the product development team and firm management can ask in order to raise the important issues and to guide the development new system with comprehensively understanding the product architecture. These questions also serve as a summary of the linkages between product architecture and strategic product development de-scribed throughout this thesis.

How to determine the design targets of new products

- Identify which attributes closely affect to customer satisfaction?
- Determine which specifications (design parameters) are related to the attributes?
- Understand the interrelationship between specifications through determining either statistical relations or physical mechanism.
- Clarify strategic constraints on specification that contain managerial directions.
- Keep eyes on the environmental changes, which can affect to customer preferences, and identify the relationship between the environmental factors and specifications.

How to compose new modules with incorporating new components

- Understand modular architecture of the existing system, and which modules should be carried over to new product. This contains identification of what kind of interactions exist between components and their characteristics.
- Identify new components, which should be incorporated to the existing system to meet the changed requirements of markets or regulations. Also, identify interfaces through which the new components will interact with the existing modules.
- Make new big module with combining the carried over modules and new components. It can be possible to make the new system higher modular and to make less change efforts to make new system.
- However, if combining cost, such as interface-standardization cost, is so high, it is recommended to leave the new components as single modules. The decision of whether composing new big module with the carried-over modules and the new components or not would purely depend on the interaction characteristics of the system or the designers' strategy.

Bibliography

Allen, K. R., and S. Carlson-Skalak. 1998. "Defining product architecture during conceptual design." ASME Design Engineering Technical Conferences, Atlanta, GA.

Alsetglobal, 2013. *Alset global's solutions* [online]. www.alset.at/our-solutions/ [Accessed 4 Nov. 2013].

Archak, N., A. Ghose, and P. G. Ipeirotis. 2011. "Deriving the pricing power of product features by mining consumer reviews." *Management Science* 57 (8): 1485-1509.

Bäck, T. 1993. "Optimal mutation rates in genetic search." International Conference on Genetic Algorithms, Citeseer.

Baldwin, C. Y., and K. B. Clark. 2000. *Design rules, volume 1: The power of modularity.* Boston: MIT Press.

Beasley, J. E., and P. C. Chu. 1996. "A genetic algorithm for the set covering problem." *European Journal of Operational Research* 94 (2): 392-404.

Benner, M. J., and M. L. Tushman. 2003. "Exploitation, exploration, and process management: The productivity dilemma revisited." *Academy of Management Review* 28 (2): 238-256.

Berry, S., J. Levinsohn, and A. Pakes. 1995. "Automobile prices in market equilibrium." *Econometrica: Journal of the Econometric Society* 63 (4): 841-890.

Booker, L. 1987. "Improving search in genetic algorithms." *Genetic algorithms and simulated annealing*: 61-73.

Campagnolo, D., and A. Camuffo. 2010. "The concept of modularity in management studies: A literature review." *International Journal of Management Reviews* 12 (3): 259-283.

Clarkson, P. J., C. Simons, and C. Eckert. 2004. "Predicting change propagation in complex design." *Journal of Mechanical Design* 126 (5): 788-797.

Cohen, L. 1995. *Quality function deployment: How to make qfd work for you.* Massachusetts: Addison-Wesley Reading.

Cooper, R. 2001. *Winning at new products: Accelerating the process from idea to launch.* 4th ed. New York: Basic Books.

De Weck, O., and M. B. Jones. 2006. "Isoperformance: Analysis and design of complex systems with desired outcomes." *Systems Engineering* 9 (1): 45-61.

De Weck, O. L., 2006. Determining product platform extent. *In* Simpson, T.W., Siddique, Z. & Jiao, J. eds. *Product platform and product family design: Methods and applications.* New York: Springer, pp. 241-301.

Eckert, C. M., M. Stacey, D. Wyatt, and P. Garthwaite. 2012. "Change as little as possible: Creativity in design by modification." *Journal of Engineering Design* 23 (4): 337-360.

Eppinger, S. D., and T. R. Browning. 2012. *Design structure matrix methods and applications.* Boston: MIT Press.

Eppinger, S. D., D. E. Whitney, R. P. Smith, and D. A. Gebala. 1994. "A model-based method for organizing tasks in product development." *Research in Engineering Design* 6 (1): 1-13.

Fixson, S. K., 2006. A roadmap for product architecture costing. *Product platform and product family design.* New York: Springer, 305-334.

Fowler, T. C. 1990. *Value analysis in design.* New York: Van Nostrand Reinhold.

Friedman, D., 2003. "*A new road: The technology and potential of hybrid vehicles."* Cambridge: Union of Concerned Scientists Publications.

Gamba, A., and N. Fusari. 2009. "Valuing modularity as a real option." *Management Science* 55 (11): 1877-1896.

Gen, M., and R. Cheng. 2000. *Genetic algorithms and engineering optimization.* New York: Wiley-Interscience Publication.

Gershenson, J., G. Prasad, and S. Allamneni. 1999. "Modular product design: A life-cycle view." *Journal of Integrated Design and Process Science* 3 (4): 13-26.

Gershenson, J., G. Prasad, and Y. Zhang. 2003. "Product modularity: Definitions and benefits." *Journal of Engineering design* 14 (3): 295-313.

Gershenson, J. K., G. J. Prasad, and Y. Zhang. 2004. "Product modularity: Measures and design methods." *Journal of Engineering Design* 15 (1): 33-51.

Giffin, M., O. De Weck, G. Bounova, R. Keller, C. Eckert, and P. J. Clarkson. 2009. "Change propagation analysis in complex technical systems." *Journal of Mechanical Design* 131 (8): 081001.

Gilbride, T. J., P. J. Lenk, and J. D. Brazell. 2008. "Market share constraints and the loss function in choice-based conjoint analysis." *Marketing Science* 27 (6): 995-1011.

Gokpinar, B., W. J. Hopp, and S. M. Iravani. 2010. "The impact of misalignment of organizational structure and product architecture on quality in complex product development." *Management Science* 56 (3): 468-484.

Green, P. E., and V. Srinivasan. 1978. "Conjoint analysis in consumer research: Issues and outlook." *Journal of consumer research* 5 (2): 103-123.

Greene, W. H. 1995. *Limdep.* New York: Econometric Software Plainville.

Gu, X., J. E. Renaud, L. M. Ashe, S. M. Batill, A. S. Budhiraja, and L. J. Krajewski. 2002. "Decision-based collaborative optimization." *Journal of Mechanical Design* 124 (1): 1-13.

Haugh, D., A. Mourougane, and O. Chatal, 2010. The automobile industry in and beyond the crisis. Economics Department: OECD.

Hauser, J. R., and D. Clausing. 1988. "The house of quality." *Harvard business review* 66 (3): 63-73.

Henderson, R. M., and K. B. Clark. 1990. "Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms." *Administrative science quarterly* 35 (1): 9-30.

Holland, J. H. 1975. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* Michigan University Press.

Horsky, D., S. Misra, and P. Nelson. 2006. "Observed and unobserved preference heterogeneity in brand-choice models." *Marketing Science* 25 (4): 322-335.

Hoyle, C., W. Chen, N. Wang, and F. S. Koppelman. 2010. "Integrated bayesian hierarchical choice modeling to capture heterogeneous consumer preferences in engineering design." *Journal of Mechanical Design* 132 (12): 121010.

Jaura, A. K., W. Ortmann, R. Stuntz, B. Natkin, and T. Grabowski. 2004. "Ford's $h_2$rv: An industry first hev propelled with an h2 fueled

engine-a fuel efficient and clean solution for sustainable mobility." *SAE SP*: 1-10.

Kamrani, A. K., and R. Gonzalez. 2003. "A genetic algorithm-based solution methodology for modular design." *Journal of Intelligent Manufacturing* 14 (6): 599-616.

Kang, C., 2010. *A balanced approach to planning and renewal of product plaforms*. Ph. D., Seoul National University.

Kang, K., and Y. S. Hong. 2013. "Re-architecting modular systems in incremental design." *Submitted to International Journal of Production Research*.

Kang, K., C. Kang, and Y. S. Hong. 2013. "Data-driven optimized vehicle-level engineering specifications." *Industrial Management & Data Systems* 114 (3).

Kim, H. M., 2001. *Target cascading in optimal system design*. PhD. Dissertation, University of Michigan.

Kwak, M. J., Y. S. Hong, and N. W. Cho. 2009. "Eco-architecture analysis for end-of-life decision making." *International Journal of Production Research* 47 (22): 6233-6259.

Lambert, A. J. D. 2002. "Determining optimum disassembly sequences in electronic equipment." *Computers & Industrial Engineering* 43 (3): 553-575.

Lancaster, J., and M. Ozbayrak. 2007. "Evolutionary algorithms applied to project scheduling problems—a survey of the state-of-the-art." *International Journal of Production Research* 45 (2): 425-450.

Lee, K. H., and I. M. Cheong. 2011. "Measuring a carbon footprint and environmental practice: The case of hyundai motors co.(hmc)." *Industrial Management & Data Systems* 111 (6): 961-978.

Lee, T. Y., and E. T. Bradlow. 2007. "Automatic construction of conjoint attributes and levels from online customer reviews." *University Of Pennsylvania, The Wharton School Working Paper*.

Maccormack, A., J. Rusnak, and C. Y. Baldwin. 2006. "Exploring the structure of complex software designs: An empirical study of open source and proprietary code." *Management Science* 52 (7): 1015-1030.

Maier, M., and E. Rechtin. 2000. *The art of systems architecting.* New York: CRC Press, Inc.

Martin, M. V., and K. Ishii. 2002. "Design for variety: Developing standardized and modularized product platform architectures." *Research in Engineering Design* 13 (4): 213-235.

Mavris, D. N., and M. R. Kirby. 1999. "Technology identification, evaluation, and selection for commercial transport aircraft." *SAWE paper*: No. 2456.

Mcfadden, D. 1974. "The measurement of urban travel demand." *Journal of public economics* 3 (4): 303-328.

Meier, C., A. A. Yassine, and T. R. Browning. 2007. "Design process sequencing with competent genetic algorithms." *Journal of Mechanical Design* 129 (6): 566-585.

Meyer, M. H., and A. P. Lehnerd. 1997. *The power of product platforms.* New York: The Free Press.

Michalek, J. J., P. Ebbes, F. Adiguzel, F. M. Feinberg, and P. Y. Papalambros. 2011. "Enhancing marketing with engineering: Optimal product line design for heterogeneous markets." *International Journal of Research in Marketing* 28 (1): 1-12.

Mikkola, J. H. 2006. "Capturing the degree of modularity embedded in product architectures." *Journal of Product Innovation Management* 23 (2): 128-146.

Newbury, S., and T. Lewin. 2008. *The car design yearbook 7: The definitive annual guide to all new concept and production cars worldwide.* New York: Merrell.

Nunez, M., V. C. Datta, A. Molina-Cristobal, M. Guenov, and A. Riaz. 2012. "Enabling exploration in the conceptual design and optimisation of complex systems." *Journal of Engineering Design* 23 (10-11): 849-872.

O'brien, R. M. 2007. "A caution regarding rules of thumb for variance inflation factors." *Quality & Quantity* 41 (5): 673-690.

Otto, K. N., and K. L. Wood. 1998. "Product evolution: A reverse engineering and redesign methodology." *Research in engineering design* 10 (4): 226-243.

Pahl, G., K. Wallace, and L. Blessing. 2007. *Engineering design: A systematic approach.* New York: Springer.

Peschka, W., E. A. Wilhelm, and U. Wilhelm. 1992. *Liquid hydrogen: Fuel of the future.* New York: Springer.

Pimmler, T. U., and S. D. Eppinger, 1994. Integration analysis of product decompositions. *Proceedings of the ASME 6th International Conference on Design Theory and Methodology.* Minneapolis, USA: Proceedings of the ASME 6th Int. Conf. on Design Theory and Methodology.

Rottengruber, H., M. Berckmüller, G. Elsässer, N. Brehm, and C. Schwarz. 2004. "Direct-injection hydrogen si-engine-operation strategy and power density potentials." *SAE paper* 113 (4): 1749-1761.

Salvador, F. 2007. "Toward a product system modularity construct: Literature review and reconceptualization." *IEEE Transactions on Engineering Management* 54 (2): 219-240.

Schaffer, J. D. 1985. "Multiple objective optimization with vector evaluated genetic algorithms." International Conference on Genetic Algorithms, 93-100 L. Erlbaum Associates Inc.

Smaling, R., and O. De Weck. 2007. "Assessing risks and opportunities of technology infusion in system design." *Systems Engineering* 10 (1): 1-25.

Sosa, M. E., S. D. Eppinger, and C. M. Rowles. 2004. "The misalignment of product architecture and organizational structure in complex product development." *Management science* 50 (12): 1674-1689.

Sosa, M. E., S. D. Eppinger, and C. M. Rowles. 2007. "A network approach to define modularity of components in complex products." *Journal of Mechanical Design* 129 (11): 1118-1129.

Sosale, S., M. Hashemian, and P. Gu. 1997. "Product modularization for reuse and recycling." *Concurrent product design and environmentally conscious manufacturing* 94: 195-206.

Steward, D. V. 1981. "Design structure system: A method for managing the design of complex systems." *IEEE Transactions on Engineering Management* 28 (3): 71-74.

Suh, E. S., 2005. *Flexible product platforms*. PhD Thesis, Massachusetts Institute of Technology.

Suh, E. S., M. R. Furst, K. J. Mihalyov, and O. De Weck. 2010. "Technology infusion for complex systems: A framework and case study." *Systems Engineering* 13 (2): 186-203.

Suh, N. P. 2001. *Axiomatic design.* London: Oxford university press.

Suzuki, T., T. Kanehara, A. Inaba, and S. Okuma. 1993. "On algebraic and graph structural properties of assembly petri net." Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, 507-514.

Tan, P. N., M. Steinbach, and V. Kumar. 2006. *Introduction to data mining.* New York: Pearson Addison Wesley.

Tate, D., D. Lindholm, and V. Harutunian. 1998. "Dependencies in axiomatic design." *Journal of Integrated Design and Process Technology* 3: 159-166.

Train, K. 2003. *Discrete choice methods with simulation.* Cambridge: Cambridge Univ Press.

Tripathy, A., and S. D. Eppinger. 2011. "Organizing global product development for complex engineered systems." *IEEE Transactions on Engineering Management* 58 (3): 510-529.

Ulrich, K. 1994. *Fundamentals of product modularity.* Netherlands: Springer Netherlands.

Ulrich, K. 1995. "The role of product architecture in the manufacturing firm." *Research Policy* 24 (3): 419-440.

Ulrich, K. T., and S. D. Eppinger. 2007. *Product design and development.* New York: McGraw-Hill.

Ulrich, K. T., and W. P. Seering. 1989. "Synthesis of schematic descriptions in mechanical design." *Research in Engineering Design* 1 (1): 3-18.

Weber, J. 2009. *Automotive development processes.* New York: Springer.

White, C., R. Steeper, and A. Lutz. 2006. "The hydrogen-fueled internal combustion engine: A technical review." *International Journal of Hydrogen Energy* 31 (10): 1292-1305.

Whitfield, R., J. Smith, and A. Duffy. 2002. "Identifying component modules." Proceedings of the 7th International Conference on Artificial Intelligence in Design, Cambridge, 571-592.

Wikipedia, 2013. *Ward's 10 best engines* [online]. Wikipedia. Available from: http://en.wikipedia.org/wiki/Ward's_10_Best_Engines [Accessed 17 May 2013].

Wyatt, D. F., C. M. Eckert, and P. J. Clarkson. 2009. "Design of product architectures in incrementally developed complex products." Proceedings of International Conference on Engineering Design (ICED 2009), Standford.

Yu, T.-L., A. A. Yassine, and D. E. Goldberg. 2003. "A genetic algorithm for developing modular product architectures." ASME.

Yu, T.-L., A. A. Yassine, and D. E. Goldberg. 2007a. "An information theoretic method for developing modular architectures using genetic algorithms." *Research in Engineering Design* 18 (2): 91-109.

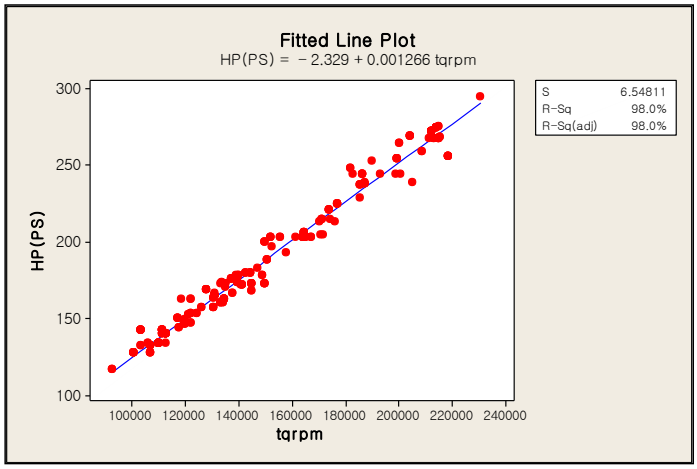
Yu, T. L., A. A. Yassine, and D. E. Goldberg. 2007b. "An information theoretic method for developing modular architectures using genetic algorithms." *Research in Engineering Design* 18 (2): 91-109.

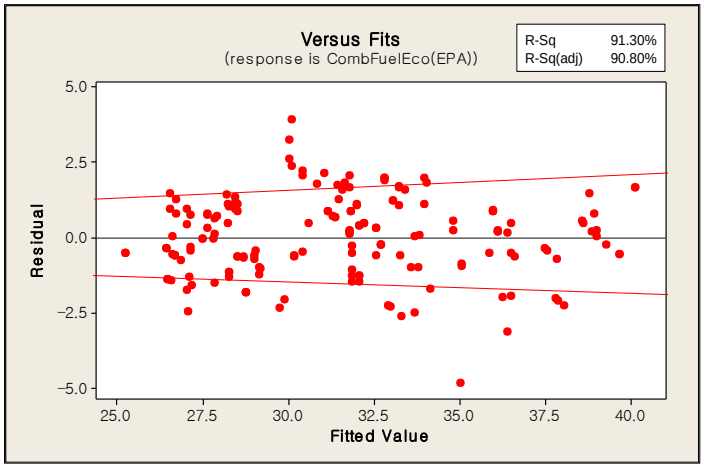
Appendix A

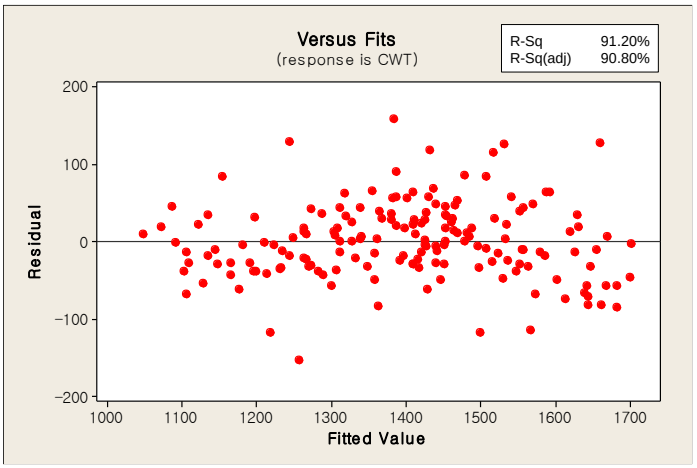
Statistical analysis of vehicle-level specifications: Elicitation of Equality and Inequality Constraints

A.1. Goodness of fit of estimated interrelations

To elicit the interrelations among the vehicle-level specifications, a regression analysis was conducted on the specification data of the historically launched vehicles. The collected data on 20 vehicle-level specifications, presented in Table #, based on 150 the mid-size sedan models on the U.S. market from 1990 to 2009.

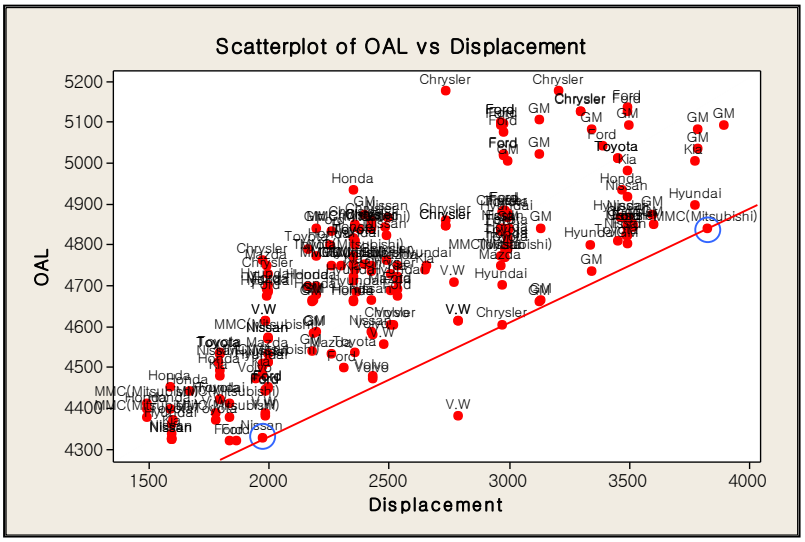

Fitted Line Plot
HP(PS) = − 2.329 + 0.001266 tqrpm
S 6.54811
R-Sq 98.0%
R-Sq(adj) 98.0%
HP(PS)
tqrpm
300
250
200
150
100
100000 120000 140000 160000 180000 200000 220000 240000

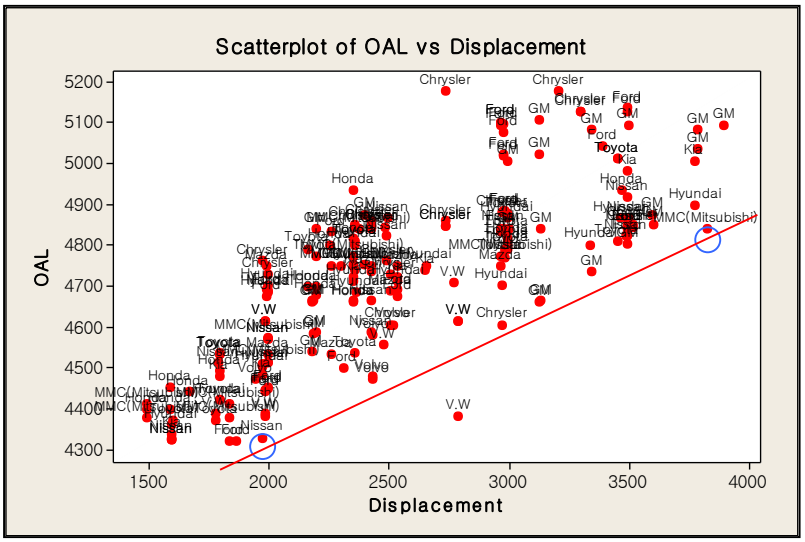

Versus Fits
(response is CombFuelEco(EPA))
R-Sq 91.30%
R-Sq(adj) 90.80%
Residual
Fitted Value
5.0
2.5
0.0
−2.5
−5.0
25.0 27.5 30.0 32.5 35.0 37.5 40.0

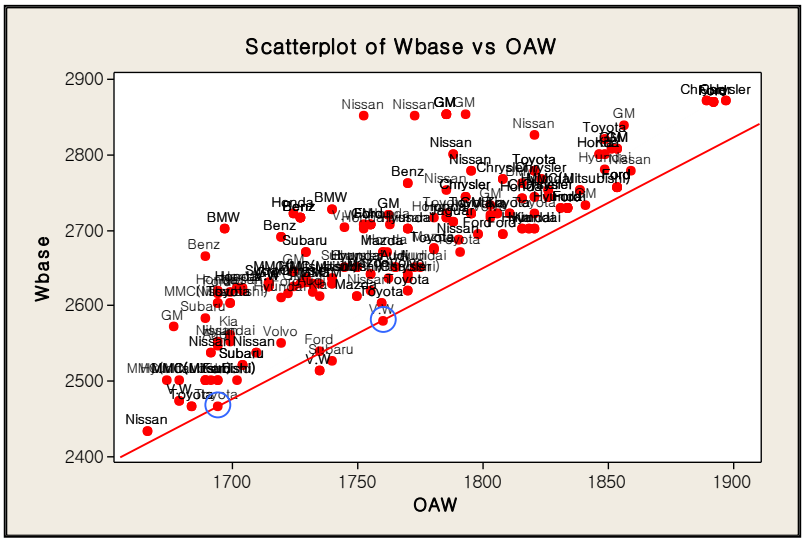

Versus Fits
(response is CWT)
R-Sq 91.20%
R-Sq(adj) 90.80%
Residual
Fitted Value
200
100
0
−100
−200
1000 1100 1200 1300 1400 1500 1600 1700

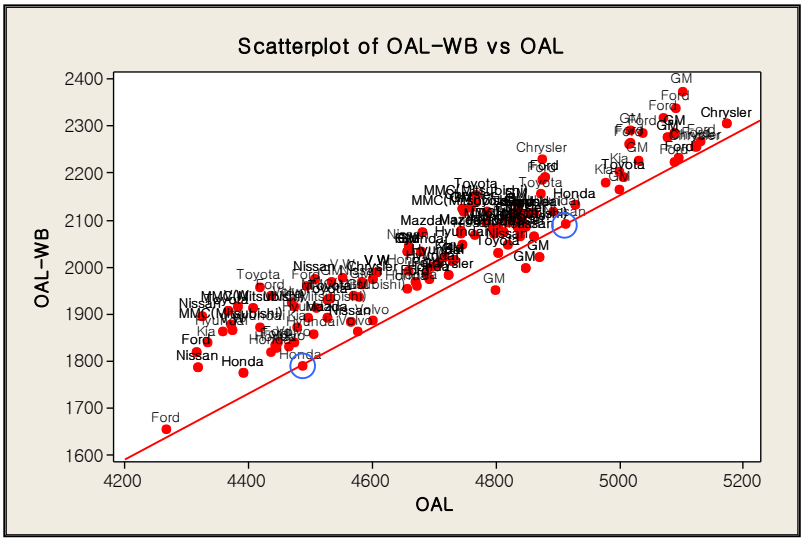
A.2. Scatter plotting for extraction of inequalities

Scatter plotting was used to figure out the boundary constraints for the vehicle-level specifications. As shown in figures following, *min* and *max* inequality constrains between correlated specifications can be elicited.






Scatterplot of Wbase vs OAW
Wbase
OAW
2900
2800
2700
2600
2500
2400
1700
1750
1800
1850
1900


Scatterplot of OAL-WB vs OAL
OAL-WB
OAL
2400
2300
2200
2100
2000
1900
1800
1700
1600
4200
4400
4600
4800
5000
5200

초 록

최근과 같이 제품이 점점 복잡해지고 각 제품의 수명주기가 극도로 짧아지는 환경에서, 창의적인 사고를 통한 신제품 개발은 실효적 측면에서 점차 그 실효성에 대한 의심을 받고 있다. 실제로, 완전히 백지로부터 시작하여 새로운 제품을 설계 혹은 개발하는 상황은 드물다. 대신, 시장의 새로운 요구 혹은 규제를 만족시키기 위해서, 거의 대부분의 신제품개발 프로젝트는 기존의 제품을 기반으로 효과적 개선 방법을 찾는데 그 초점을 맞추고 있다. Eckert et al. (2012)는 많은 제품개발 프로젝트가 처음부터 새로운 제품을 설계하는 방식으로 이루어지기 보다는 기존의 제품들을 수정하는 방향으로 진행된다고 밝혔다. 이러한 신제품개발 방법에서는 이미 여러 번의 시험과정을 통해 그 성능이 입증된 기존의 시스템을 최대한 활용함으로써, 예상치 못한 위험과 개발비용을 최소화할 수 있다.

본 연구에서는 새로운 요구기준을 만족하기 위해 기존 시스템을 수정 혹은 재설계 하는 활동을 ‘증분설계(incremental design)’라고 정의하고, 그것을 위한 효과적인 전략을 수립하는 방법론을 제시한다. 기존 제품을 기반한 신제품 개발이 성공하기 위해서는 기존제품의 아키텍처를 명확히 이해하고, 그것을 바탕으로 설계 목표를 효과적으로 설정하여, 실제 제품으로 구현하는 것이 필요하다. 제품 아키텍처(product architecture)는 제품이 수행하여야 하는 기능과 제품을 구성하는 물리적 컴포넌트간의 연결관계로, 기존제품의 아키텍처는 증분설계 상황에서 신제품에 대한 일종의 설계제약으로 작용하게 된다. 신제품 개발을 위한 설계 목표를

설정하기 위해서, 기존 제품이 수행해 오던 기능 간의 관계를 파악하고, 이를 기반으로 새로운 요구를 최대한 충족시킬 수 있는 제품 사양을 결정한다. 이렇게 결정된 설계목표는 물리적 컴포넌트를 통해서 실현 되는데, 이때 신기술이 적용된 새로운 컴포넌트가 기존 시스템에 삽입되어 새로운 시스템을 구성한다. 오랜 시간 안정화된 기존제품의 물리적 아키텍처는 새로 도입하는 컴포넌트로 인해 변경이 불가피 하게 된다. 따라서, 신제품을 위한 컴포넌트 간의 아키텍처의 재정립이 필수적이다.

증분설계 상황에서 설계목표를 결정하기 위해 본 연구에서는 기존 제품의 아키텍처를 기능면에서 정의하고, 그 아키텍처를 기반으로 시장에서 요구하는 제품 사양을 결정하는 방법론을 제시한다. 제품의 기능은 제품의 사양으로 표현되는데, 본 연구에서는 사양간의 상호관계를 이용하여 기존 제품의 아키텍처를 정의하였다. 사양간 상호관계는 시장에 출시된 제품들의 사양정보를 통계적으로 분석하여 추출하였고 이는 곧 설계가능공간의 정의라고 할 수 있다. 시장에서 판매된 제품들은 이미 그들의 사양자체에 실현가능성을 내포하고 있기 때문이다. 한편, 제품의 사양은 제품의 속성이 되어 소비자에게 제품의 가치를 전달하게 된다. 정의된 기존 제품의 아키텍처를 기반으로 소비자에게 가장 높은 효용을 제공하는 제품 사양을 결정함으로써, 실현가능하면서 소비자에게 높은 효용을 제공할 수 있는 설계 목표를 설정할 수 있다. 다수의 마케팅 연구에서는 소비자 선호를 평가하여 제품을 기획하는 방법을 제안해 왔지만, 실제로 기획한 제품이 실현가능한지에 대해서는 고려하지 못했다. 한편, 공학설계의 분야에서는 시장에서부터 오는 소비

자 선호정보를 설계 변수에 연계시키지 못한 채, 실현 가능성에 집중한 나머지 시장성이 있는 제품을 설계하는데 한계점이 있었다. 본 연구에서 제안하는 방법론은, 시장으로부터 오는 정보와 설계정보 간의 유기적 상호작용을 체계적으로 포착하여, 최종적 설계목표를 설정하는데 효과적으로 활용하였다는 데 공헌이 있다.

결정된 설계 목표는 물리적 컴포넌트를 통해서 실현된다. 증분설계 상황에서는 기존의 시스템에 새로운 컴포넌트가 도입되어 새로운 시스템을 구성하게 된다. 이때 기존의 제품이 모듈설계를 통해서 효율적 아키텍처를 구축하고 있는 상황이라 하면, 새로운 컴포넌트는 전체 시스템의 아키텍처 변경(change)을 유발한다. 본 연구는 기존의 제품을 기반으로 새로운 제품을 개발하는 증분설계 상황에서, 기존시스템에 최소한의 변경을 가하며, 높은 모듈성을 가진 아키텍처를 구성하는 방법론을 제안한다. 제안하는 아키텍처재정립 방법론은 기존제품의 존재를 명확하게 인정하고, 기존의 아키텍처가 신제품을 위한 최적아키텍처로 변환해 가는 과정자체에 집중한다. 기존의 모듈설계 및 플랫폼 구성과 관련한 다수의 연구들이 신제품을 위한 최적아키텍처를 찾는 방법론을 제안하였다. 이 방법론들은 기존시스템의 존재를 내재하면서, 최종해를 찾는데 집중하였다. 하지만, 기존의 시스템으로부터 최적아키텍처에 도달하는 방법은 매우 다양할 수 있으며, 사실 어떻게 기존의 아키텍처를 새로운 요구에 맞추어 잘 변형시켜가는가에 따라 최적 아키텍처가 달라질 수도 있다. 따라서, 본 연구는 기존의 아키텍처로부터 다양한 변형을 통해 신제품을 위한 후보아키텍처를 생성하는 과정을 모형화하고, 이를 바탕으로 최적 아키텍처를 선정한다.

주요어: 증분설계, 제품아키텍처, 설계전략, 설계제약, 아키텍처재정립, 모듈설계
학번: 2007-20876