Proceedings of the 19th CIRP Design Conference – Competitive Design, Cranfield University, 30-31 March 2009, pp456

# Combining Axiomatic Design and Case-Based Reasoning in a Design Methodology of Mechatronics Products

N. Janthong[1,2,3], D. Brissaud[1], S. Butdee[2]
[1]G-SCOP research laboratory, University of Grenoble, France
[2]Integrated Manufacturing System Research Center (IMSRC), Department of Production Engineering, Faculty of Engineering, King Mongkut's University of Technology North Bangkok (KMUTNB), Thailand
[3]Thai-French Innovation Centre (TFIC), King Mongkut's University of Technology North Bangkok (KMUTNB), Thailand

**Abstract**
Current market environments are volatile and unpredictable. The ability for design products to meet customer's requirements has become critical to success. The key element to develop such products is identifying functional requirements and knowledge utilization based on a scientific approach to provide both designers of new products and redesigners of existing products with a suitable solution that meets to customer's needs. This paper presents a method to (re)design mechatronic products by combining the axiomatic design and case-based reasoning approaches. Innovation has increased the new product value, which has improved the product efficiency and the need for new engineered design method.

**Keywords:**
Engineering design; Adaptable design; CBR; Axiomatic design

## 1 INTRODUCTION

Mechatronic is a technology which combines mechanics with electronics and information technology to form both interaction and spatial integration in components, modules, products and systems [1]. In fact all electronically controlled mechanical systems are based on the idea of improving products by adding features from other types of products. The result is that new product functionality is created and more efficient technologies utilized. These caused by industrial circumstances change and existing product with its function is no longer satisfied. Thus, existing product life will be extended. In industrial environment, however, customers need specific machines to perform specific tasks in their industry, which their functions and performances may be different or similar to the previous generation. In this sense, customers' needs have become very personalized and the major factor to guide the development of such products. However, the success of new product development in satisfying one customer goes through the reuse of elements of the responses to previous customers. By reusing previous designs, an engineer can reduce duration and cost of development cycle and risks on product quality and performances. Moreover, the relevant and innovative information in any design discipline may also be mobilized and used to update or adapt a previous design in response to changes in technology or market preferences.

As show in figure 1, the reuse of design is normally needed for some modification or adaptation, which can occur in two ways.

The first one is the adaptation of a previous product of the product family to a new list of requirements; the advantage is that the producer can adapt the same design to different requirements and produce different product models.

The second one is the adaptation across product families. The advantage comes from reusing the same 'design' for different product families, and lead to the ability of sharing functions and components.

Hence, organizing, storing and retrieving information on previous product designs are the most important tasks in knowledge utilization to provide both designers of new products and redesigners of existing products with a suitable solution that meets the customer's needs.

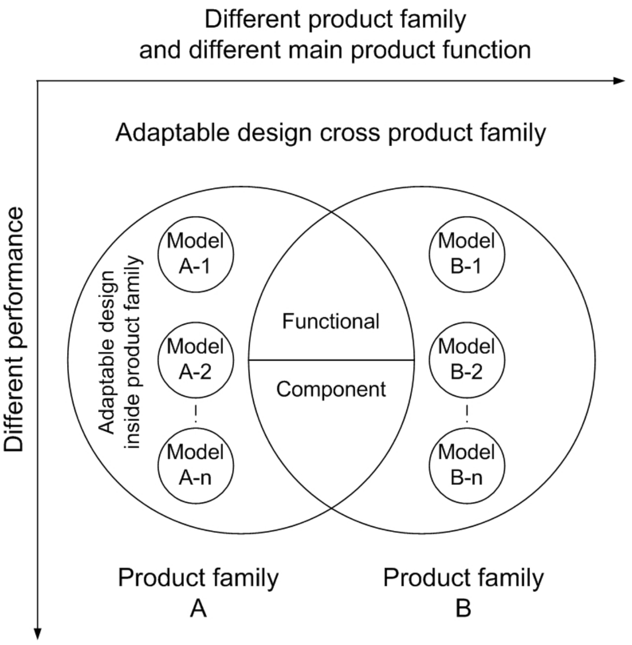


Figure 1: The reuse of design output

Our objective is to develop a new methodology to support the design of mechatronic products based on principles of generating new ideas from both new knowledge and previous solutions based on company experience while minimizing risks. Section 2 will present the literature review then section 3 will develop the methodology proposed and applied on the example of a redesigned electric vehicle as automated vehicle. Conclusions are given in section 4.

## 2 LITERATURE REVIEW

The main technique reviewed was case-based reasoning (CBR) applied to design. The basic idea of case-based reasoning is that new problems can be tackled by adapting solutions that were used to solve previous problems [2, 3]. Case-Based Reasoning is a general paradigm for problem solving based on the recall and reuse of experience. The practice shows that it is often more efficient to solve a problem by starting from a solution of a previous, similar problem than to generate the entire solution from scratch. Due to the mentioned properties, CBR systems have a multitude of applications in architecture design [4], in chemical process engineering [5, 6], in injection mould design [7], and in mechanical design [8, 9] as well as design for mass customization [10] etc. The two major research issues in the CBR approach to design are the representations of design cases and the process model for recalling and adapting design cases [11]. Representing design cases requires an abstraction of the experience into a symbolic form that the system can manipulate [12, 13]. Design-case recalling involves finding a relevant design experience: it is broken down into the subtasks of indexing, retrieval, and selecting. Indexing design cases is a critical issue in CBR approach, and CBR systems suffer from an inability to distinguish between cases if indexing is inadequate [14]. Design-case adaptation recognizes the differences between the selected design case and the new design problem, and changes the design case so that it solves the new design problem. This process is divided into three steps: propose, evaluate, and modify.

As the literature review showed, case-based reasoning techniques have been investigated and the principles and technology are now mature. The concept of case based reasoning can be defined in the way to organize information or data, and this concept is applied to either 'idea', innovation or any other kinds of information that is to be stored and used at a later point in time.

However, mechatronic products designs are especially difficult to represent as a well-structured list of features. The representations of design cases require various models of knowledge from each domain. Highly structured representations of design knowledge can be used for reasoning. However, case-based reasoning usually require manual pre or post processing, structuring and indexing of design knowledge to identify the information needed by designers. There is a need to develop a method that clearly determines mechatronic products design requirements. One such a method that rigorously defines the design requirements could be Axiomatic Design.

Axiomatic Design defines design as the creation of synthesized solutions in the form of products, processes or systems that satisfy perceived needs through mappings between Functional Requirements (FRs) and Design Parameters (DPs) [15]. The implementation issues were discussed by many publications [16, 17, 18, 19, 20, 21]. A fundamental aspect of the mapping process is the idea of break down through zigzagging. The design progresses from a higher, abstract level down to a more detailed level. This result in the formation of design hierarchies in the FRs and DPs which are similar in nature to standard product functional and structural hierarchies. Thus it can identify which parts of the design structure are used to perform specific functions.

To facilitate (re)designing mechatronics product, this paper combines the axiomatic design and case-based reasoning approaches. The case based reasoning is used as a general framework for the reuse of product designs and applied when a similar function is required. The axiomatic design principle is used for creating cases by analyzing existing products which FRs and DPs were decomposed. These FRs and DPs are utilized as case index and case representation in case libraries. It is also used for creating design databases or design libraries by identifying relationships between FRs and possible DPs of each component in a design library. The information content is used for evaluating design solutions (DPs) from design libraries or design databases composed of various components to fulfill a new functional requirement which is not yet existing in the case libraries. The design with satisfy independence axiom provides the sequence to modify DPs in the adaptation process.

## 3 DESIGN METHODOLOGY

The methodology as shown in figure 2 is based on the assumption that the designers do not need to design products from scratch every time. They go through their ability to access to existing designs from related products and components, then revise them to fulfill specific customers' needs. Figure 3 is an example of real world problems that we solved based on the concepts afore mentioned. The function structure and the physical structure of products from past design experiences were stored in a case library. Moreover, the design library kept the designs which included components information and their function definition, which come from supplier's standard catalogue. Both the case library and the design library were utilized to create suitable design solutions to achieve the new functionalities. Reuse case when new customers' requirements have similar function combines with new design sub-functions when retrieved case doesn't have function that customer wanted. It is the basic concept for combining case based reasoning and Axiomatic design principles. The process started by the comparison of new customers' requirements and constraints to function structures and physical structures of existing products that perform similar requirements and constraints. The result is that functions can be separated in product functions that have already been developed in existing designs, and add-on functions that did not exist and require to be fulfilled through the designs process. To achieve that, add-on functions are decomposed in terms of functional requirements; physical solutions are retrieved by comparison to other products of the family and by searching in design databases and standard components libraries. The retrieval process based on functionality and other specifications is accomplished by the aid of an inference engine. Both rules and cases are necessary for the reasoning process. Then, adaptations of the design is needed to re-configure and integrate components to achieve the new design. Thus, product architecture, platforms, modules as well as functional and physical structures are the main drivers to create the case base. The adaptation process needs to follow the most suitable sequence.

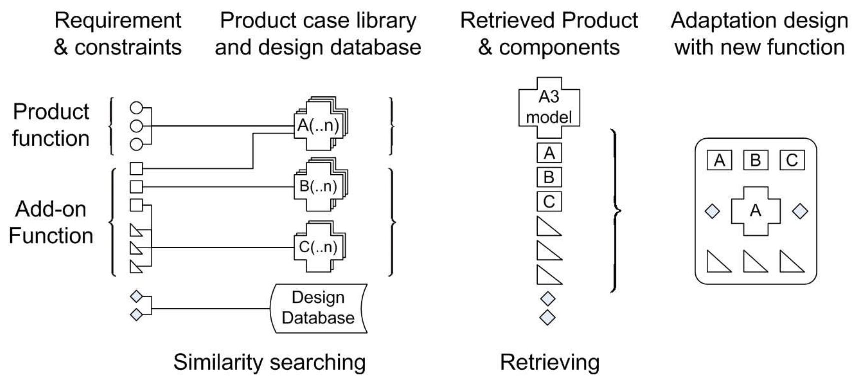

Requirement & constraints
Product case library and design database
Retrieved Product & components
Adaptation design with new function
Product function
Add-on Function
A(..n)
B(..n)
C(..n)
Design Database
A3 model
A
B
C
Similarity searching
Retrieving

Figure 2: Design Methodology

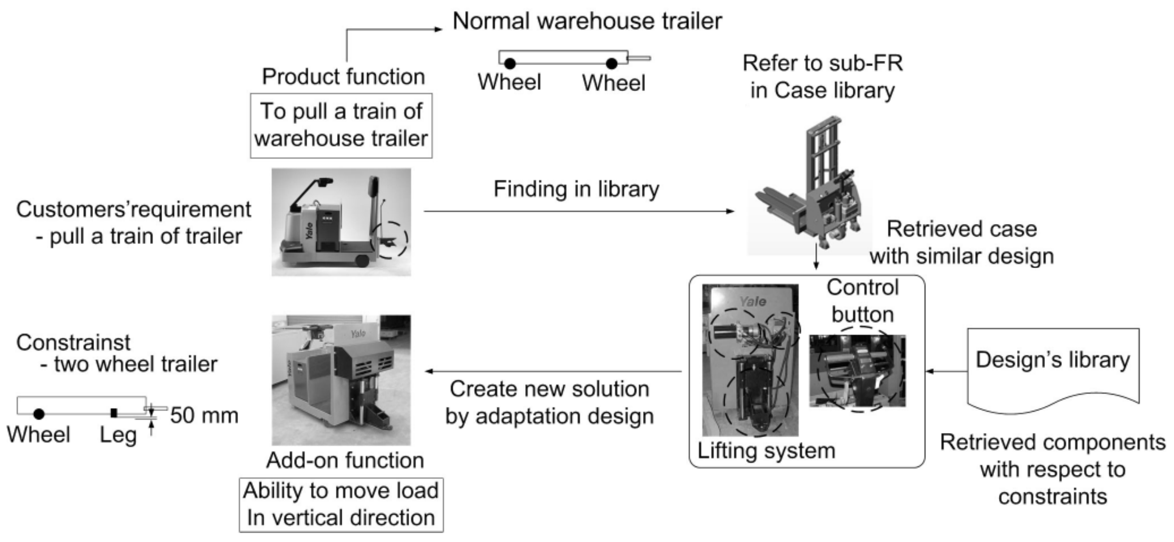

Normal warehouse trailer
Product function
To pull a train of warehouse trailer
Wheel
Wheel
Refer to sub-FR in Case library
Customers'requirement - pull a train of trailer
Finding in library
Retrieved case with similar design
Control button
Constrainst - two wheel trailer
Wheel
Leg
50 mm
Create new solution by adaptation design
Design's library
Lifting system
Add-on function
Ability to move load In vertical direction
Retrieved components with respect to constraints

Figure 3: An example of add-on functions

### 3.1 Case Representation

The basic idea is to organize specific cases, which share similar properties under a general structure. The scheme of a case consists of four parts as shown in figure 4, including customer's requirements, customers' constraints, functional requirements, and design parameters. The case represented in terms of a design hierarchy in each of the domains: functional and physical. The hierarchical structure in the FR-domain and the DP-domain correspond to customers' requirements. An advantage of this representation is that it allows a case to be accessed on its whole or by its parts when a new problem must be solved. Similar cases at appropriate levels of abstraction are retrieved from the case base and the solutions from these cases are combined and refined; the constraints can be used to guide adaptation.

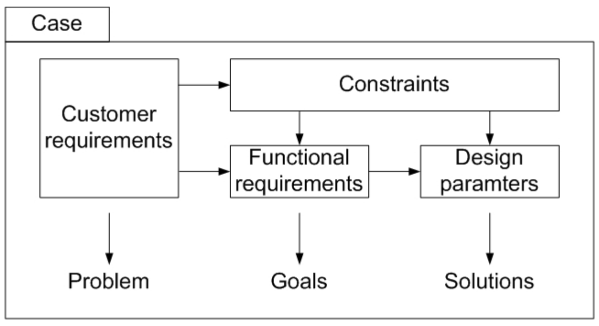


Figure 4: The scheme of a case representation

### 3.2 Case Indexing

Case indexing involves assigning indices to cases for their quick and easy retrieval from a case library. Axiomatic design decomposition principles are used to determine the indexing of both design cases and their solutions as shown in figure 5. A hierarchical case library is similar in nature to the product architecture; designers often care of design the entire systems down to the lowest component levels that compose the systems. Thus, cases are indexed by their functions allowing a case to be retrieved in several ways. This indexing structure scheme also allows the composition of different case pieces to create a new solution.

However, it usually require manual pre or post processing, structuring and indexing of design knowledge to identify the information needed by designers. Based on axiomatic design principle, designers map from the requirements what they want the design do to the solutions of how the design will achieve these. As the design progresses, broad, high-level requirements are broken down into smaller sub-requirements, which are then satisfied met by sub-solutions. It is also important to maintain the functional independence. That is why the index structure was created to distinguish between the cases in the case library.

An example of the index structure and solutions of high lift stacker product is shown in figure 6. It shows that there are many different ways to satisfy the FRs. FR skeleton sets can be generated for each of the design cases in the case library. Each step down the hierarchy represents a refinement of the unit design. It helps distinguish between cases which lead to efficient case matching and retrieving. This example comes from past design experiences satisfying customers' needs and the underlying product architecture in product family. This expresses that firms can manage single products and platforms to deliver the different products while sharing components.

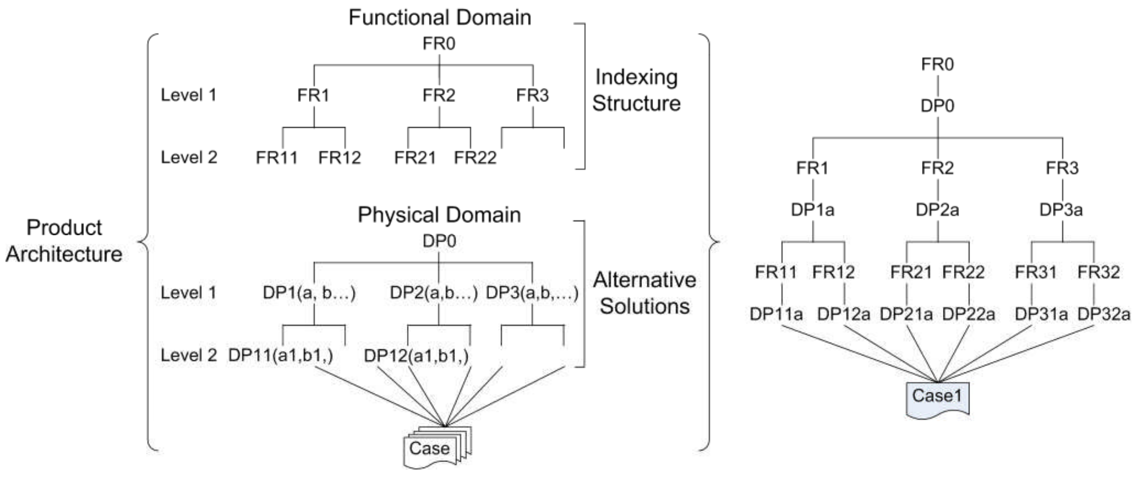
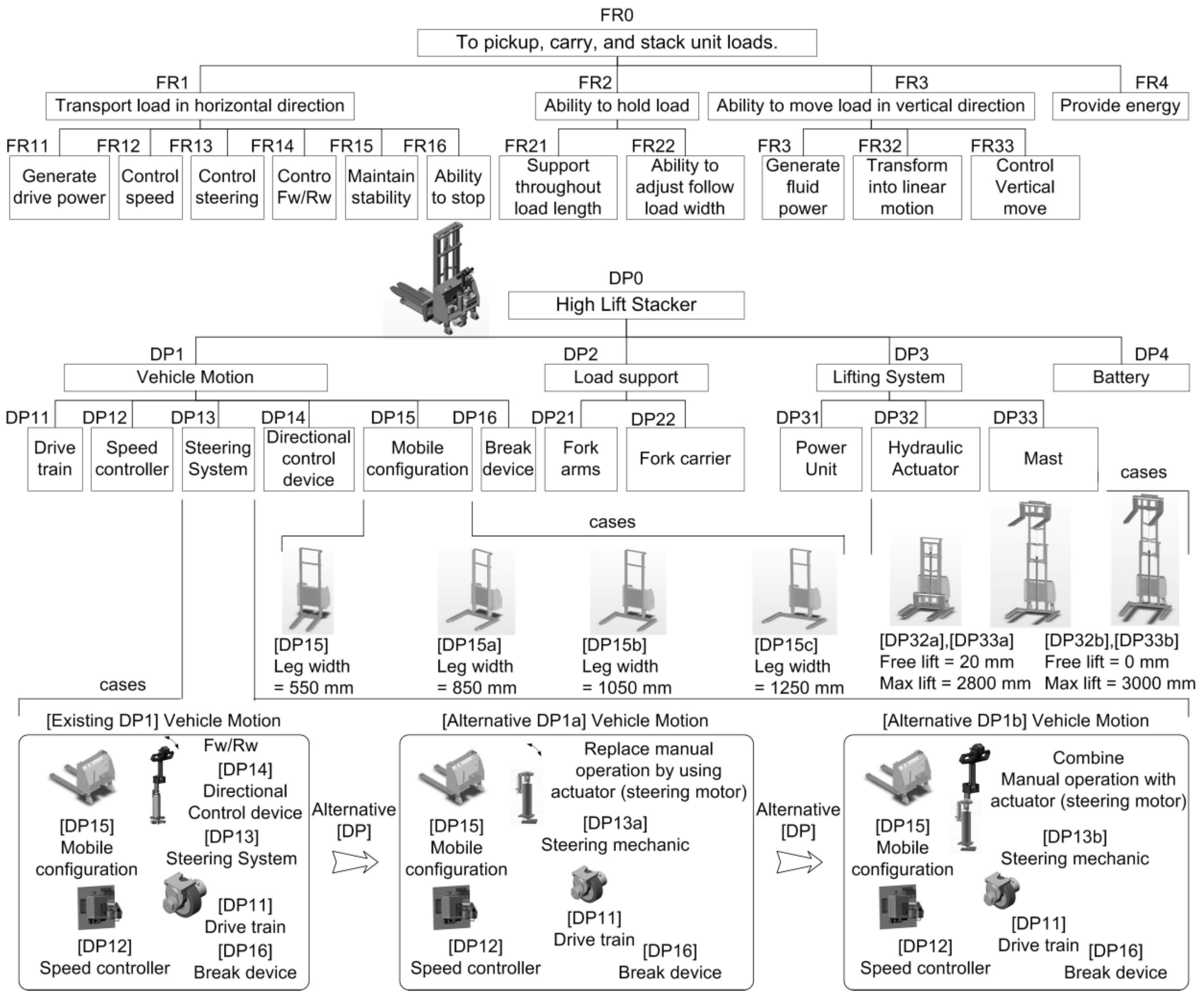


Figure 5: An index structure in case library



FR0
To pickup, carry, and stack unit loads.
FR1
Transport load in horizontal direction
FR2
Ability to hold load
FR3
Ability to move load in vertical direction
FR4
Provide energy
FR11
Generate drive power
FR12
Control speed
FR13
Control steering
FR14
Contro Fw/Rw
FR15
Maintain stability
FR16
Ability to stop
FR21
Support throughout load length
FR22
Ability to adjust follow load width
FR3
Generate fluid power
FR32
Transform into linear motion
FR33
Control Vertical move
DP0
High Lift Stacker
DP1
Vehicle Motion
DP2
Load support
DP3
Lifting System
DP4
Battery
DP11
Drive train
DP12
Speed controller
DP13
Steering System
DP14
Directional control device
DP15
Mobile configuration
DP16
Break device
DP21
Fork arms
DP22
Fork carrier
DP31
Power Unit
DP32
Hydraulic Actuator
DP33
Mast
cases
cases
[DP15] Leg width = 550 mm
[DP15a] Leg width = 850 mm
[DP15b] Leg width = 1050 mm
[DP15c] Leg width = 1250 mm
[DP32a],[DP33a] Free lift = 20 mm Max lift = 2800 mm
[DP32b],[DP33b] Free lift = 0 mm Max lift = 3000 mm
cases
[Existing DP1] Vehicle Motion
Fw/Rw
[DP14] Directional Control device
[DP15] Mobile configuration
[DP13] Steering System
[DP11] Drive train
[DP12] Speed controller
[DP16] Break device
Alternative [DP]
[Alternative DP1a] Vehicle Motion
Replace manual operation by using actuator (steering motor)
[DP15] Mobile configuration
[DP13a] Steering mechanic
[DP11] Drive train
[DP12] Speed controller
[DP16] Break device
Alternative [DP]
[Alternative DP1b] Vehicle Motion
Combine Manual operation with actuator (steering motor)
[DP15] Mobile configuration
[DP13b] Steering mechanic
[DP11] Drive train
[DP12] Speed controller
[DP16] Break device

Figure 6: An example index structure of modified high lift stacker

### 3.3 Case retrieval

As afore mentioned, when new customers' requirements and constraints are given, similar historical design cases are searched, matched and retrieved. The result is that two major functions are classified, namely product functions from existing products in case library that have similar functions according to problem inputs and add-on functions that are not on the retrieved existing products. Thus, the case retrieval process includes two phases – (i) similarity matching of product functions and (ii) similarity matching of add-on functions. Each phase relies on achieving two goals: finding a similar case set and finding the most similar case in this set.

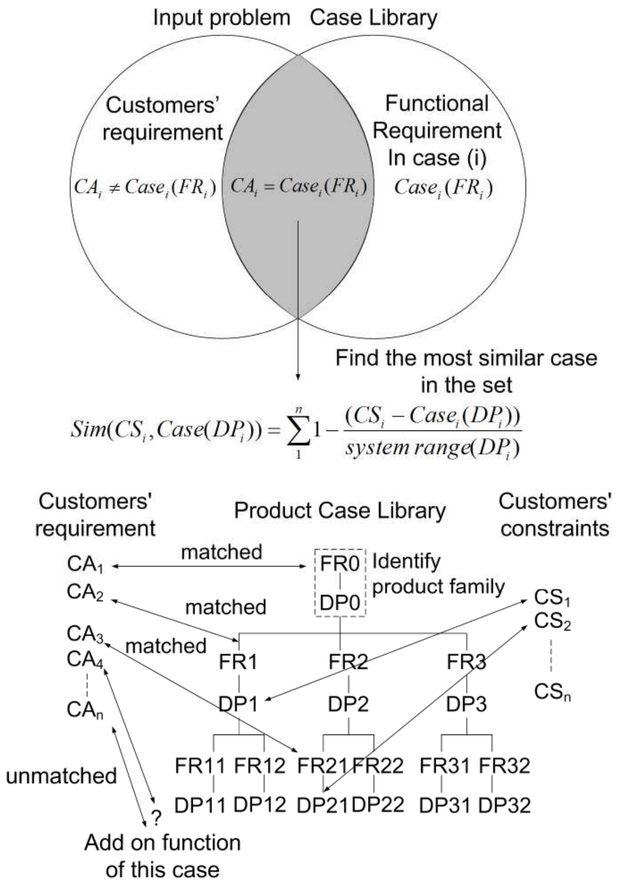


Figure 7: Case retrieval based on similarity of product function matching

In the first phase, the similarity matching of product functions as shown in figure 7, finds the similar case set from customers' requirements *(CA$_i$)* that are compared to product function hierarchy of each case *(Case$_i$(FR$_i$))*. The simplest similarity measure is to score 1 for equality and 0 for inequality as follow:

$$sim(CA_i, Case_i(FR_i)) = \begin{cases} 1 & if\ CA_i = Case_i(FR_i); \\ 0 & otherwise \end{cases}$$

Thus, a set of cases from the case base that are similar to the current input case is equal to the intersection of *(CA$_i$)* and *(Case$_i$(FR$_i$))* as follows :

$$\{P\} = CA_i \cap Case_i(FR_i) \quad ; \{P\} = set\ of\ relevant\ product$$

After all similar cases are found, a mechanism to find the most similar case in this set is needed. The input constraints *(CS$_i$)* are used to compare to design parameters of each retrieved case. Then, the $sim(Cs_i, Case_i(DP_i))$ can be calculated by:

$$sim(Cs_i, Case_i(DP_i)) = \sum_1^n 1 - \frac{(Cs_i - Case_i(DP_i))}{system\ range(DP_i)}$$

where: *Cs$_i$ - Case$_i$(DP$_i$)* is the difference between the feature values of the input and the retrieved case and *system range(DP$_i$)* is the range which each DP can satisfy FR based on the capacity of the producer. Then to turn a normalized distance function into a similarity measure, its value subtracts from 1. The set of cases are ranked by these similarity scores and retrieves case with the highest similarity score.

In the second phase, the add-on functions (as shown in figure 8) are the customer's attributes *(CA$_i$)* that did not match *(Case$_i$(FR$_i$))* in the first phase. There are two possibilities for the remaining CAs. The first one is when the function does not exist in the retrieved cases but could be in other product families: the system is called to search in the other product libraries by the same procedure as the first phase.

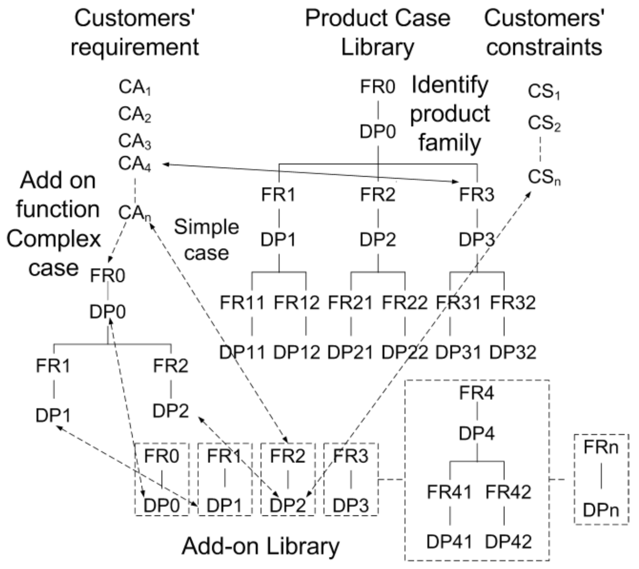


Figure 8: Case retrieval based on similarity of add-on functions matching

The second one is when the function does not exist at all in any cases of the database: the producer never did this function before for any product they did. A new design of the function of the product must be created. The add-on library and the designs database include mechanical parts, electrical parts, software modules etc. These add-on components are defined as pairs of FR and DP for single component and hierarchy of FR and DP in case of assembly components.

Similar to matching *CA$_i$* with *Case$_i$(FR$_i$)* in the first phase, *Design$_i$(FR$_i$)* are defined to distinguish the sources of information between case library for reuse design and design library for new design. The components in design database were evaluated to find solution which satisfies add-on function. If a CA corresponds to one function, a solution can be found by ***sim(CA$_i$,Design$_i$(FR$_i$))***. If a CA corresponds to a hierarchy of functions we can find the solution of the CAs by sim***(CA$_i$(FR$_i$),Design(FR$_i$))***. Thus, matching a *CA$_i$* with ***Design$_i$(FR$_i$)*** is as follows :

$$sim(CA_i, Design_i(FR_i)) = \begin{cases} 1 & if\ CA_i = Design_i(FR_i); \\ 0 & otherwise \end{cases}$$

or

$$sim(CA_i(FR_i), Design_i(FR_i)) = \begin{cases} 1 & if\ CA_i(FR_i) = Design_i(FR_i); \\ 0 & otherwise \end{cases}$$

The solution to satisfy each FR must be evaluated by minimizing the information content of the design based on Suh's axiomatic design principle as follows:

$$I = \log_2 \frac{system\ range}{common\ range}$$

$$I_{total} = \sum_1^n I_i$$

The idea mentioned above is shown in figure 9. In the general case-base system, cases and designs experience are usually used to solve new problems by evaluating similar cases and modifying or adapting the retrieved cases. In our work, we found that designing is a complex task and it is unreasonable to expect a case base to contain all the possible design cases. It is the reason why our methodology combines case-based reasoning to initiate an appropriate design due to past experience and axiomatic design rules to provide this design with the new functions needed. This provides a combined advice that is better to satisfy design constraints and compatibility requirements compared to only CBR system.

An example of the retrieval of a towing vehicle case is shown in figure 10. The example shows that the matching first retrieved the towing vehicle product then retrieved the component from high lift stacker satisfying the function.

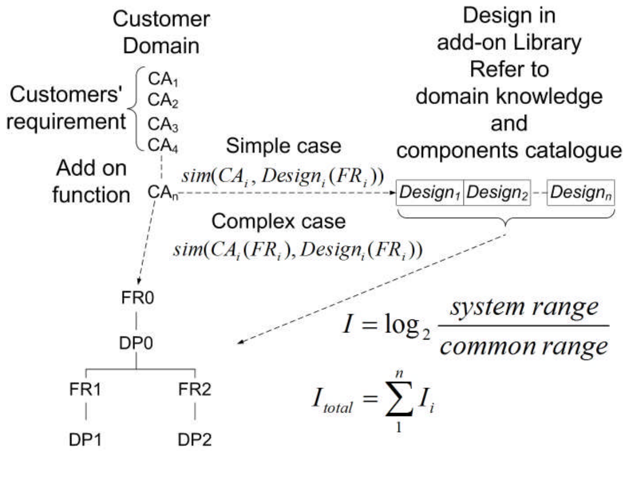

Customer Domain
Design in add-on Library Refer to domain knowledge and components catalogue
Customers' requirement
CA1
CA2
CA3
CA4
Add on function
CAn
Simple case
sim(CA_i, Design_i(FR_i))
Design1
Design2
Designn
Complex case
sim(CA_i(FR_i), Design_i(FR_i))
FR0
DP0
FR1
FR2
DP1
DP2
I = log2 system range / common range
I_total = Σ I_i

Figure 9: The design retrieval concept

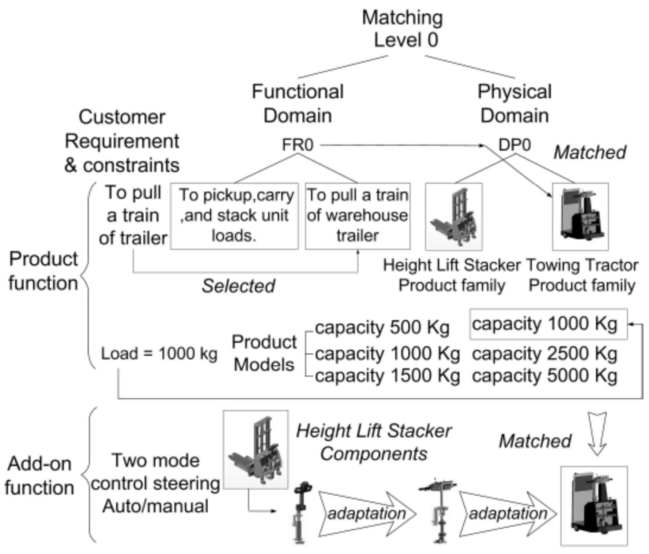

Matching Level 0
Functional Domain
Physical Domain
Customer Requirement & constraints
FR0
DP0
Matched
To pull a train of trailer
To pickup,carry ,and stack unit loads.
To pull a train of warehouse trailer
Product function
Selected
Height Lift Stacker Product family
Towing Tractor Product family
Load = 1000 kg
Product Models
capacity 500 Kg
capacity 1000 Kg
capacity 1500 Kg
capacity 1000 Kg
capacity 2500 Kg
capacity 5000 Kg
Add-on function
Two mode control steering Auto/manual
Height Lift Stacker Components
Matched
adaptation
adaptation

Figure 10: An example or retrieved product model and add-on components

## 3.4 Case Adaptation

If an exact matching case is found from the case retrieval process, its design can be used for the new order without any modification. Otherwise, and adaptation process is invoked to detect the discrepancies between the most similar case and the new order, and to reconcile the discrepancies by adapting the past design to the new situation.

The adaptation knowledge is usually represented as rules. The adaptation rules specify, under a certain situation, how to modify the value of a feature, or how to insert or delete certain features of the case representation in order to generate a solution for the new problem. According to axiomatic design principles when the relationships between FRs and DPs is uncoupled design, the set of adaptation rules can be easily and automatically selected by the system to make effect on similar old case and to produce the new one. Uncoupled design occurs when each FR is satisfied by exactly one DP. The resulting matrix is diagonal and the design equation has an exact solution. The selection of adaptation rules is done easily by comparing the conflicting differences between the new problem and the current retrieved case.

In addition, the sequence of applying the adaptation rules is also important because when the design matrix is lower triangular the resulting design is decoupled, which means that a sequence exists, where the FRs can be satisfied by adjusting DPs in a certain order. This is a very important finding, as the design process is determined to a great extent by this sequence.

Figure 11 shows the simple case to express the application of the concept mentioned above. Axiomatic design is applied in case adaptation process of the customized leg of the high lift stacker (customers' requirement comes from the size of the pallet). The resulting matrix is lower triangular; the resulting design is decoupled. It means that the adaptation rules first need to adjust parameter DP1 to achieve FR1 and then adjust parameter DP2 to satisfy FR2. If the case adaptation process does not follow the sequence specified by the triangular design matrix, the system appears to be very complex, which is defined as the imaginary complexity [22].

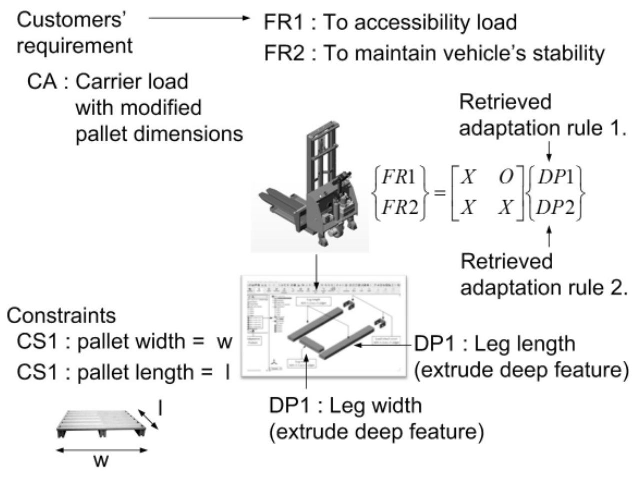

Customers' requirement
FR1 : To accessibility load
FR2 : To maintain vehicle's stability
CA : Carrier load with modified pallet dimensions
Retrieved adaptation rule 1.
$\begin{Bmatrix} FR1 \\ FR2 \end{Bmatrix} = \begin{bmatrix} X & O \\ X & X \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \end{Bmatrix}$
Retrieved adaptation rule 2.
Constraints
CS1 : pallet width = w
CS1 : pallet length = l
DP1 : Leg length (extrude deep feature)
DP1 : Leg width (extrude deep feature)
l
w

Figure 11: An example of applying axiomatic design principle in the case adaptation process

While the traditional approach of case based reasoning does not specify how to consider the sequence to adjust parameters of old case features, so there is no clear way to guarantee the correct sequence to apply adaptation rules. The axiomatic design principle can help designers make decisions in order to adapt old cases to solve new cases without a random manner to satisfy the desired system function.

## 4 CONCLUSIONS AND FURTURE WORK

This paper has presented the concept of combining axiomatic design and case-based reasoning, to assist the design process of evolving systems of industrial products. The paper illustrates how companies can react to customers' demand of industrial products in very competitive market. The company knowledge can be stored, then reused and integrated with various technologies from design databases and generate new functionalities for improving the existing products. With this methodology, the customer can extend existing product's life (refer to machine they already used) and the producer can provide customized products which have new functionalities suited customer's needs (refer to ability to create a variety of product).

Currently a software implementation based on this methodology is being developed. It consisted in formalizing the case in two main parts: the problem (customers' requirement and constraints) and the product (functional requirements, design parameters and components), to create the library of cases and design database. The problem was formalized in an adequate manner enable the calculation of the similarity function. The product was formalized to highlight the driven parameters of the design.

Two critical processes in case based reasoning were addressed namely case retrieval process and case adaptation process. The critical process in case retrieval process is functions classification, namely product functions when existing products in case library have similar functions according to input problem, and add-on functions when the retrieved existing product does not have such function. In addition, case adaptation process is most important to achieve reusability past designs for new situation. Axiomatic design principle is the systematic approach in engineering design which can assist design engineers to design products and also case based systems. Function and physical decompositions are the basic method to represent cases and are also used to define indexes in case library as well as used to determine new designs when no case exists in case library. It also supports design engineers to achieve the adaptable design by the defined sequence of the adaptable process when the design is decoupled.

However, the quality of the design solution depends on the set of FRs and DPs in the case and add-on library. Design engineers must carefully decompose the set of FRs and DPs in existing product functions and add-on component library functions that will be further reused. The next step in this project is to address the adaptation process by extending the system with knowledge on the global behavior of the product to fulfill the (re)design of mechatronic products.

## 5 REFERENCES

[1] Xu, Y. and Huijun, Z., "Function principles for a mechatronics system design", J. Engineering Manufacture Proc. IMechE, Vol.201, 2007,pp 1065-1077.

[2] Aamodt, A. and Plaza, E., "Case-based reasoning: Foundational issues methodological variations, and system approaches", AI Communications, Vol 7, 1994, pp. 39-59.

[3] Watson, I., "Case-based reasoning is a methodology not a technology", Knowledge-Based Systems, Vol 12, 1999, pp. 303-308.

[4] Heylighen, A. and Neuckermans, H., "A case base of Case-Based Design tools for architecture", Computer-Aided Design, Vol 33, 2001, pp.1111-1122.

[5] Suh, S. M., Jhee C.W., Ko K.Y., and Lee., "A case-based expert system approach for quality design", Expert Systems with Application, Vol 15, 1998, pp.181-190.

[6] Avramenko, Y. and Kraslawski, A., "Similarity concept for case-based design in process engineering", Computer & Chemical Engineering, Vol 30, 2006, pp.548-557.

[7] Hu, W. and Masood, S., "An Intelligent Cavity Layout Design for Injection Moulds", International Journal of CAD/CAM, Vol.2, No.1, 2002, pp. 69-75.

[8] Qin, X. and William C. R., "Applying Case-Based Reasoning to Mechanical Bearing Design", In Proc. Of the ASME 2000 DETC conferences, #DETC2000/DFM-14011, 2000.

[9] Vong C.M., Leung, T.P. and Wong, P.K., "Case-based reasoning and adaptation in hydraulic production machine design", Engineering Application of Artificial Intelligence, Vol 15, 2002, pp.567-585.

[10] Tseng, M.M. and Jiao, A, "Case-Based Evolutionary design for mass customization", Computers and Industrial Engineering, Vol. 33, No. 1-2, 1997, pp. 319-324.

[11] Maher, M.L., and A. Gomez de Silva Garza, "Case-Based Reasoning in Design", IEEE Expert, 1997,pp. 34-41.

[12] Praehofer, H. and Kerschbaummayr, J, "Case-based reasoning techniques to support reusability in a requirement engineering and system design tool", Engineering Application of Artificial Intelligence, Vol.12, 1999, pp. 717-731.

[13] Han, Y.H., and Lee, K., "A case-based framework for reuse of previous design concepts in conceptual

synthesis of mechanisms", Computers In Industry, Vol. 57, 2006, pp. 305-318.

[14] Boyle, M.I. and Rong, K., " CAFIXD: A Case-Based Reasoning Fixture Design Method. Framework and Indexing Mechanisms" ,In Proc. Of the ASME 2000 DETC/CIE conferences, #DETC2004-57689, 2004.

[15] Suh, N.P., The Principles of Design, Oxford University Press, 1990.

[16] Harutunian, V., Nordlund, M., Tate, D., and Suh, N.P., Decision Making and Software Tools for Product Development Based on Axiomatic Design Theory, Annals of the CIRP, Vol.45/1, 1996. pp.135-139.

[17] Suh, N.P., Design of Systems, Annals of the CIRP, Vol.46/1, 1997, pp.75-80.

[18] Suh, N.P., and Do, S.H., Axiomatic Design of Software Systems, Annals of the CIRP, Vol.49/1, 2000.pp.95-100.

[19] Melvin, J.W., and Suh, N.P., Simulation within the Axiomatic Design Framework, Annals of the CIRP, Vol51/1, 2002.pp.107-110.

[20] Deo, H.V., and Suh, N.P., Mathematical Transforms in Design: case Study on Feedback Control of a Customizable Automotive Suspension, Annals of the CIRP, Vol53/1, 2004,pp.125-128.

[21] Goncalves-Coelho, A.M., and Mourao, J.F., Axiomatic design as support for decision-making in a design for manufacturing context : A case study., International journal of production economics, Vol.109, 2007, pp. 81-89.

[22] Suh, N.P., Complexity in Engineering, Annals of the CIRP, Vol53/1, 2004.