Journal of Engineering Science and Technology
Vol. 11, No. 1, 2016, pp.1-17

MULTI-OBJECTIVE GROUPING GENETIC ALGORITHM FOR PRODUCT LIFECYCLE OPTIMISATION

MICHAEL J. LEE[1], KEITH CASE[1]* & RUSSELL MARSHALL[2]

[1]Mechanical and Manufacturing Engineering, Loughborough University
Loughborough, Leicestershire, LE11 3TU, UK
[2]Loughborough Design School Loughborough University
Loughborough, Leicestershire, LE11 3TU, UK
*Corresponding Author: K.Case@lboro.ac.uk

Abstract

A product's lifecycle performance (e.g. assembly, outsourcing, maintenance and recycling) can often be improved through modularity. However, modularisation under different and often conflicting lifecycle objectives is a complex problem that will ultimately require trade-offs. This paper presents a novel multi-objective modularity optimisation framework; the application of which is illustrated through the modularisation of a car climate control system. Central to the framework is a specially designed multi-objective grouping genetic algorithm (MOGGA) that is able to generate a whole range of alternative product modularisations. Scenario analysis, using the principles of the analytical hierarchical process (AHP), is then carried out to explore the solution set and choose a suitable modular architecture that optimises the product lifecycle according to the company's strategic vision.

Keywords: genetic algorithms, life cycle design, multi- criteria decision making, optimization, modularity.

1. Introduction

To design and manufacture successful and profitable products in an increasingly competitive consumer-based economy presents many manufacturing companies with significant challenges. In most industries product development times are becoming shorter as companies are forced to offer an increasing number of new products to the market in order to remain competitive. In the automotive industry for example product development times for new products have shrunk dramatically in the last two decades, and this trend is set to continue well into the foreseeable future [1]. In addition, many companies now have complex global supply chains, where product design and manufacture is outsourced to various suppliers around the world. Tighter environmental legislation is also having effects on the way products are developed - in some industries the producer now has to ensure their products are able to meet strict recycling and reuse targets. In order for companies to remain profitable under these demanding conditions a number of strategic design and manufacture strategies are sought. One such important strategy is the use of well-defined modular product architectures.

Modular product architecture is a means by which a product can be decomposed into smaller, more manageable chunks, to improve various lifecycle performance characteristics. However, creating a suitable modular product architecture that optimises many lifecycle considerations is a complex task: it is often impossible to simultaneously optimise the modular architecture across the whole product lifecycle as some of these goals may be conflicting. The overall aim of the research presented in this paper has therefore been to develop a multi-objective optimisation framework for product modularisation. In the framework numerous product modularity principles have been reconciled and a state-of-the-art multi-objective optimisation algorithm has been developed to perform module grouping. A software prototype of the framework has been implemented using Visual Basic within an excel environment. The main focus of this paper is to describe the framework steps using a case study example – a car climate control system.

Modularity has been given many definitions over the years and a large range of measures, methods and techniques have been created in the attempt to guide the development of modular product architectures. Generally speaking one can see a kind of general convergence towards the seminal works of Ulrich and Tung [2], who define modularity in terms of two characteristics of product design: similarity between the physical and functional architecture of the design and the minimisation of incidental interactions between physical components.

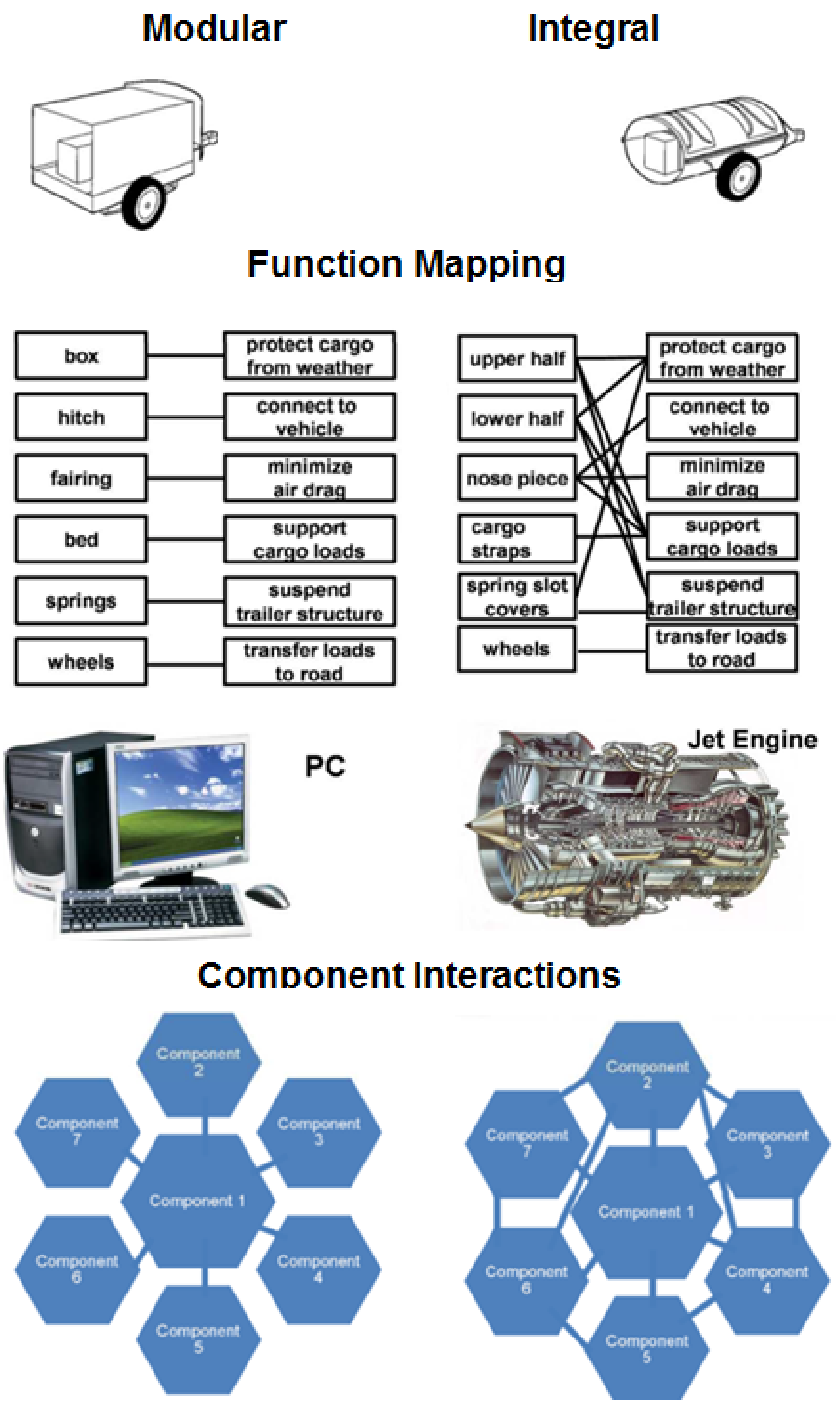


Fig. 1. Modular vs Integral Architecture (adapted from [3])

The first part of Ulrich and Tung’s [1] definition, relates product modularity to product functions. Similar perspectives on modularity can be seen in other works [5-11]. Following these works, product architecture is defined by the way in which functional elements correspond to physical components (Fig. 1).The product architecture is said to be modular when it exhibits a one-to-one mapping between functional and physical elements. In a modular product functions are less integrated (spread among components), so different customer needs can be addressed by different modules, allowing a mix and match of modules to enable product variety at low costs [2, 10, 12]. The second part of Ulrich and Tung's seminal definition of modularity views modularity in terms of the interaction and interface complexity (coupling) between components. Other works also support this idea [13-21]. Interactions can be seen as the physical and functional relationships between the product's elements (components). Obviously, there is a need to reduce the number and complexity of these interactions between modules. This will reduce design dependencies, reduce assembly complexity and can be used in the pursuit of 'plug-in - plug-out' or inter-changeable modules to create a large number of product variants at low cost. Other researchers have chosen to include other product lifecycle based aspects into their definitions of modular products. Some authors [16] view modularity from a whole lifecycle viewpoint, and the methods have been used in pursuit of service [22, 23], manufacturing [24, 25], retirement [26] and assembly [27, 28] based modularity and green engineering [29, 30]. Similarly, other bodies of work [14, 15, 31, 32] also see modularity as a means of improving various product lifecycle goals.

In regards to actually creating a modular product, there have been numerous frameworks and methods developed to create optimal modular product architectures. The majority of these methods pursue a 'bottom-up' approach in which low-level product elements (components) are grouped to form larger product element (modules). The rationale for grouping has been seen to vary considerably, from a more technical perspective such as functional and physical interactions to a more strategic focus such as the similarity between various lifecycle attributes such as service and reliability, reuse and recycling, product variety, outsourcing, etc. A clustering approach based on functional interactions between components has been used [33]. Single objective mathematical optimisation models have also been developed; such as [15] who have developed a heuristic and non-linear optimisation model to optimise a weighted sum of numerous lifecycle objectives. Similarly [24] use a non-linear optimisation genetic algorithm (GA) based weighted sum approach to modularise to a number of strategic modular drivers and functional/ physical interactions, and [35] have also developed a GA-based clustering method. They apply a Module Strength Indicator (MSI) which results in an alternative representation of the design structure matrix (DSM) useful for identification of a modular hierarchy within the product structure.

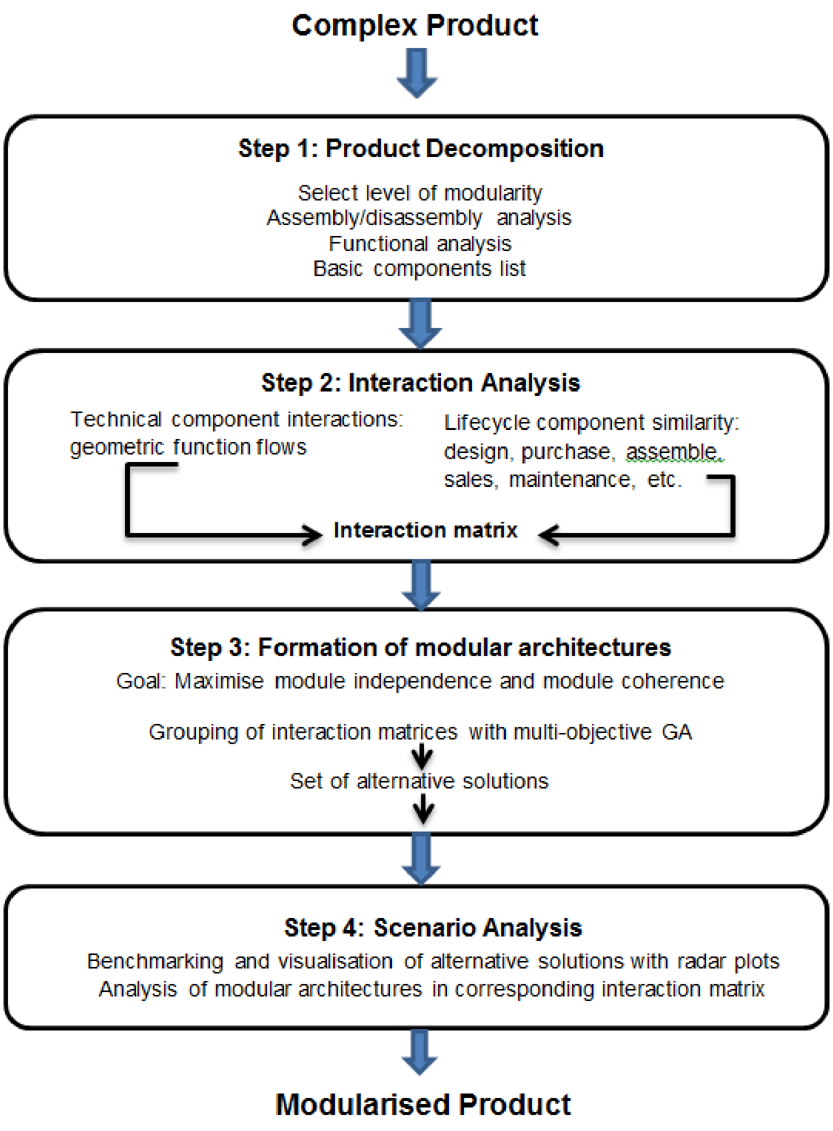
Manual heuristic based methods have also been developed. Modular Function Deployment (MFD) [36] uses a comprehensive list of modular drivers which can be used to evaluate candidate modules. Functional models have been created from time ordered function chains and use a set of heuristics to form modules [5]. A matrix based approach has been used [37] that provides fuzzy logic based interaction evaluation to integrate four sets of sub-objectives into four modularity performance goals (cost, maintenance and reliability, quality and manufacturability) that are then handled by a goal programming based optimisation model.

From the above literature review it is clear that numerous modularisation methods and frameworks have been created, often pursuing similar lifecycle goals. The majority of these methods are matrix based, using a DSM approach to represent the complex functional, physical, and strategic-based interactions that occur between components. Matrix representations are a highly visual way of representing product modularity and, more importantly, can be readily manipulated with optimisation algorithms to identify modules. However, it is argued that existing product modularisation frameworks take a simplistic approach to the multi-objective nature of the modularity configuration problem and that finding an optimal solution (where trade-offs may be needed between conflicting objectives) with these approaches can be problematic and time-consuming.

2. Overview of the modularisation Framework

The developed modularity optimisation framework has four main steps: 1) product decomposition; 2) interaction analysis; 3) formation of modular architectures; and; 4) scenario analysis (Fig. 2). The developed framework has been implemented within a software prototype using Microsoft Excel and Visual Basic, and the main user interface screen can be seen in Fig. 3. In summary of the framework steps; the product is first decomposed into a number a basic components by analysis of product functionally and existing physical structure. The various interfaces and lifecycle similarities that occur between the product's components are then analysed and entered into a DSM-style interaction matrix. In the developed software implementation of the framework, a specially developed multi-objective grouping genetic algorithm (MOGGA) then searches the matrix and provides a whole set of alternative (Pareto-optimal) modular product configurations. The solution set is then evaluated and explored (scenario analysis) using multi-criteria decision making (MCDM). In the developed software prototype the Analytic Hierarchy Process [38] approach has been implemented for this final step.

It must be noted that even for a relatively simple product there could be many different modular configurations; each optimised for different lifecycle goals. With each different configuration there will of course be compromises that will have to be made between the different modularisation objectives e.g. the modular structure may be easy to assembly, yet have poor maintainability and recyclability. Ideally these compromises should be explored before arriving at a final decision. The important aspect of the framework is the novel multi-objective approach to product modularisation, in which a whole set of alternative modular product architectures are generated in one single run, for further analysis with MCDM. This provides two main advantages. Firstly, the user does not have to provide objective preference weights. Secondly, time-consuming preference adjustment and algorithm re-runs are not needed in order to preform scenario analysis.



Complex Product
Step 1: Product Decomposition
Select level of modularity
Assembly/disassembly analysis
Functional analysis
Basic components list
Step 2: Interaction Analysis
Technical component interactions: geometric function flows
Lifecycle component similarity: design, purchase, assemble, sales, maintenance, etc.
Interaction matrix
Step 3: Formation of modular architectures
Goal: Maximise module independence and module coherence
Grouping of interaction matrices with multi-objective GA
Set of alternative solutions
Step 4: Scenario Analysis
Benchmarking and visualisation of alternative solutions with radar plots
Analysis of modular architectures in corresponding interaction matrix
Modularised Product

Fig. 2. Overview of modularity framework

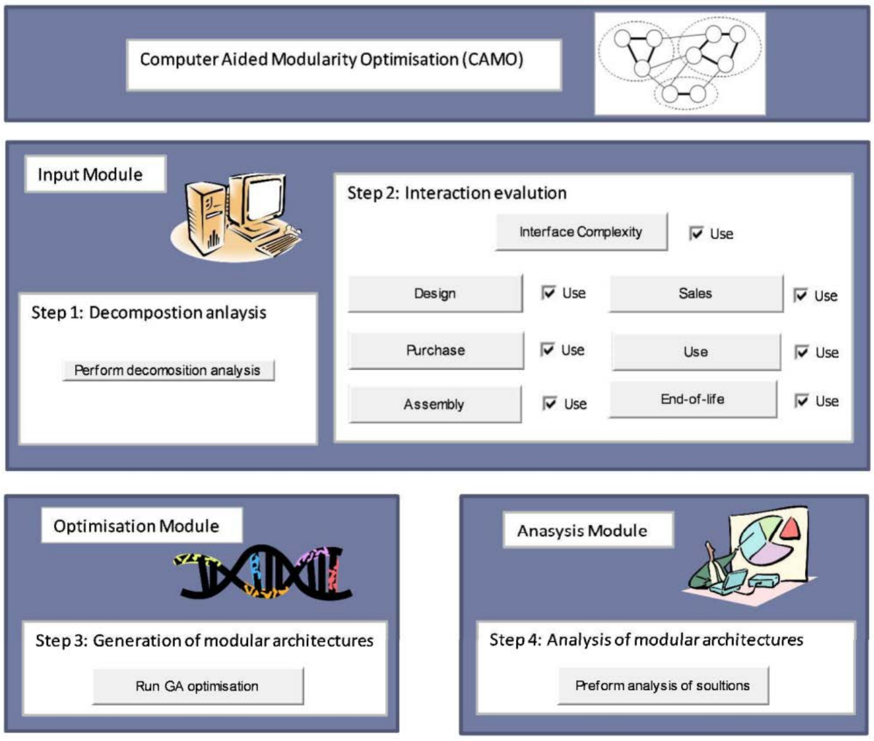

Computer Aided Modularity Optimisation (CAMO)
Input Module
Step 2: Interaction evalution
Interface Complexity
Use
Design
Use
Sales
Use
Purchase
Use
Use
Use
Assembly
Use
End-of-life
Use
Step 1: Decompostion anlaysis
Perform decomosition analysis
Optimisation Module
Step 3: Generation of modular architectures
Run GA optimisation
Anasysis Module
Step 4: Analysis of modular architectures
Preform analysis of soultions

Fig. 3. Main user interface of software prototype implementation of modularity framework

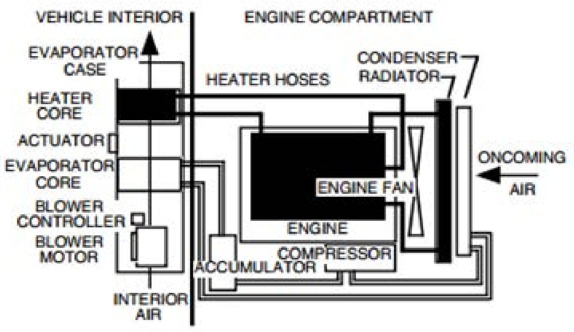
3. Product Modularisation with the Framework

In this section the various steps of the framework are described using an example product: the car climate control system. The car climate control system is well referred in other studies [33, 37, 39]; it is a fairly complex product, comprised of various technologies that must be split across numerous geometric locations within the car, making it an ideal case example to assess the potential of the developed modularity framework.

3.1. Step 1: Product decomposition

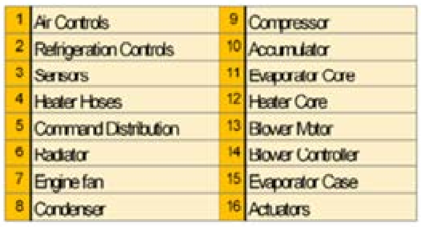
Essentially with the framework we are decomposing a product into its main functional components which are then grouped into modules based upon their interface complexity and lifecycle similarity. However, just like existing hierarchical product assemblies, it is argued that modularity may also exist at many levels within the product. Furthermore, complex products may contain hundreds/thousands of components and it would be infeasible to attempt modularisation of such large components sets. The decomposition step should therefore aim to identify the main functional components at a suitable level in the product hierarchy. This may mean that a functional ‘component’ is in fact a complete subassembly/ assembly. To aid the decomposition process established methodologies are recommended (e.g. function-means trees [40] or reverse fish bone diagrams [31]). The physical decomposition of the automotive climate control system (Figs. 4a and 4b) follows Pimmler and Eppinger's work [33], who decompose the system into 16 main functional components (Fig. 4c).

(a)



(b)

(c)



1	Air Controls	9	Compressor
2	Refrigeration Controls	10	Accumulator
3	Sensors	11	Evaporator Core
4	Heater Hoses	12	Heater Core
5	Command Distribution	13	Blower Motor
6	Radiator	14	Blower Controller
7	Engine fan	15	Evaporator Case
8	Condenser	16	Actuators

Fig. 4. a) Diagram of car climate control system [33] b) illustration of car climate control system within car (adapted from [41])

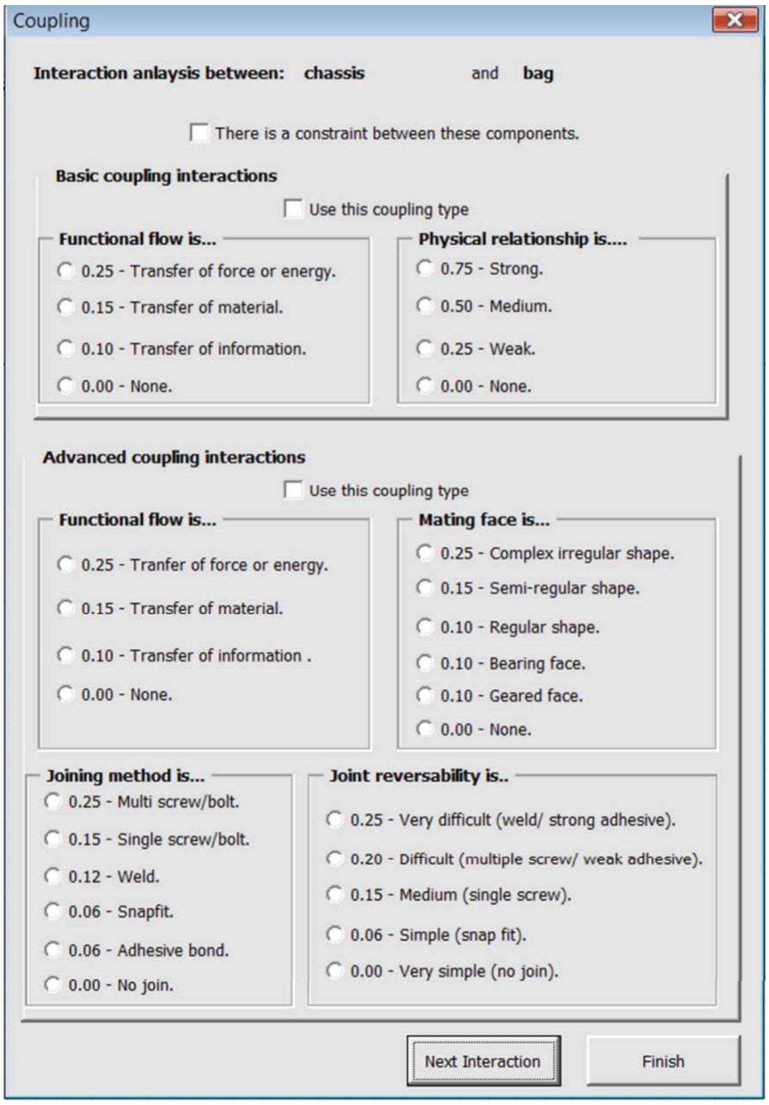
c) Main components of the automotive climate control system

3.2. Step 2: Interaction analysis

Once the basic physical components have been identified, the physical and functional interactions between components and the lifecycle similarities of components are analysed and entered into a master interaction matrix. In the developed software prototype, to assist the interaction analysis, evaluation forms, such as for example of coupling interactions (Fig. 5), have been developed using VBA for excel. There are two types of information that must be entered into the matrices: the physical and functional interactions between components and the lifecycle similarity of components.

3.2.1. Component coupling analysis

Following the principles of Ulrich and Tung [2] components that have strong physical and functional relationships should be grouped together to ensure that modules are as independent (loosely coupled) as possible. This will simplify interface complexity between modules and help reduce assembly and disassembly costs. The approach advocated in this research is to use one of two component interface evaluations. If the product is in the conceptual stages where the joining methods have not yet been chosen or the mating complexity between components is unknown it is suggested that a simpler measure (basic component coupling analysis) is used for component interface analysis (Fig. 5). For this evaluation the user must enter the level of component coupling due to three types of functional flows as well as the estimated degree of physical coupling. If sufficient knowledge is known about the possible joining methods and the level of mating complexity then the advanced coupling evaluation can be used. This is based upon three sub factors: the joining method, the mating face complexity and the interface reversibility between components. The combined interface value is then entered into the corresponding position of the matrix. For the purpose of this case study example basic coupling interactions are used, which follow the physical and functional interactions of Pimmler and Eppinger's work.


Coupling

Interaction anlaysis between: chassis and bag

☐ There is a constraint between these components.

Basic coupling interactions

☐ Use this coupling type

Functional flow is...
- ○ 0.25 - Transfer of force or energy.
- ○ 0.15 - Transfer of material.
- ○ 0.10 - Transfer of information.
- ○ 0.00 - None.

Physical relationship is....
- ○ 0.75 - Strong.
- ○ 0.50 - Medium.
- ○ 0.25 - Weak.
- ○ 0.00 - None.

Advanced coupling interactions

☐ Use this coupling type

Functional flow is...
- ○ 0.25 - Tranfer of force or energy.
- ○ 0.15 - Transfer of material.
- ○ 0.10 - Transfer of information .
- ○ 0.00 - None.

Mating face is...
- ○ 0.25 - Complex irregular shape.
- ○ 0.15 - Semi-regular shape.
- ○ 0.10 - Regular shape.
- ○ 0.10 - Bearing face.
- ○ 0.10 - Geared face.
- ○ 0.00 - None.

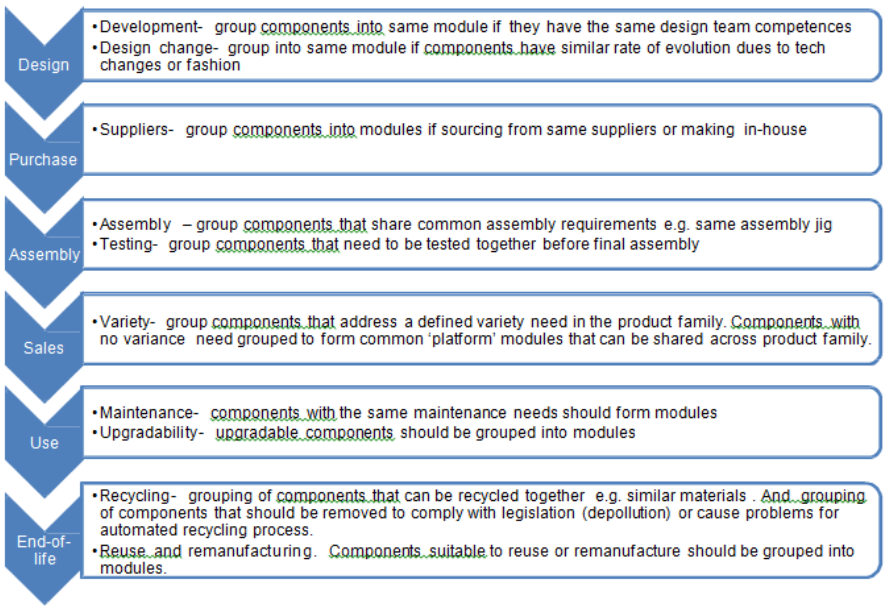
Joining method is...
- ○ 0.25 - Multi screw/bolt.
- ○ 0.15 - Single screw/bolt.
- ○ 0.12 - Weld.
- ○ 0.06 - Snapfit.
- ○ 0.06 - Adhesive bond.
- ○ 0.00 - No join.

Joint reversability is..
- ○ 0.25 - Very difficult (weld/ strong adhesive).
- ○ 0.20 - Difficult (multiple screw/ weak adhesive).
- ○ 0.15 - Medium (single screw).
- ○ 0.06 - Simple (snap fit).
- ○ 0.00 - Very simple (no join).

Fig. 5: Interaction evaluation form for loose coupling objective

3.2.2. Component lifecycle similarity

In this framework, to optimise product lifecycle characteristics, components that have the same lifecycle requirements should be grouped into discrete modules. There have been many lifecycle modularity drivers identified in the literature, such as the comprehensive list provided by Erixon [26]. In order to provide the basis for a multi-objective optimisation these drivers have been classified into six different product lifecycle stages (Fig. 6). In the developed software prototype, evaluation of the component lifecycle similarity between components is done by splitting the associated modular drivers into a number of different nodes which are mapped to components. This module driver mapping is done in a binary manner; a ‘1’ is entered into the appropriate column if the component corresponds to that driver (Fig. 7). For example, for the climate system, the modularity driver of product variety is split into 4 different nodes based upon different product variance and commonality needs. Each node thus represents a possible module boundary for that particular lifecycle phase. If purely optimised for variance it is clear that there would be 4 modules.



Design
- Development- group components into same module if they have the same design team competences
- Design change- group into same module if components have similar rate of evolution dues to tech changes or fashion

Purchase
- Suppliers- group components into modules if sourcing from same suppliers or making in-house

Assembly
- Assembly – group components that share common assembly requirements e.g. same assembly jig
- Testing- group components that need to be tested together before final assembly

Sales
- Variety- group components that address a defined variety need in the product family. Components with no variance need grouped to form common ‘platform’ modules that can be shared across product family.

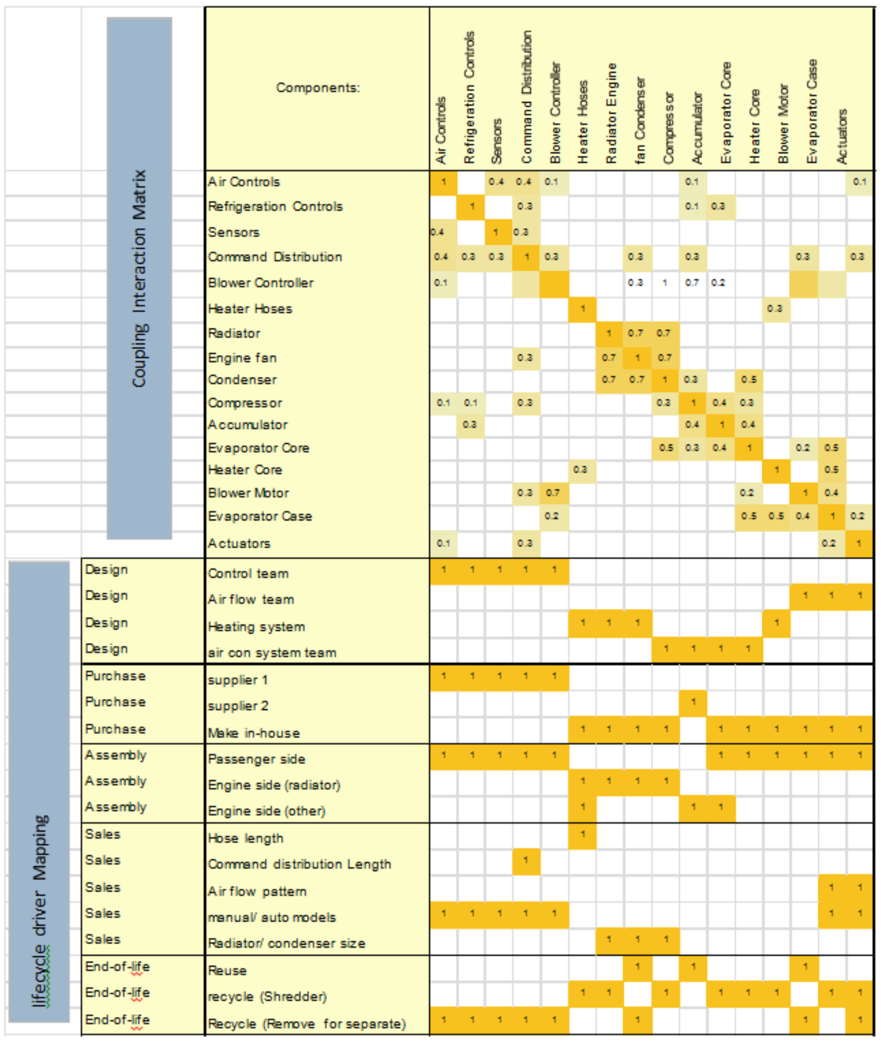
Use
- Maintenance- components with the same maintenance needs should form modules
- Upgradability- upgradable components should be grouped into modules

End-of-life
- Recycling- grouping of components that can be recycled together e.g. similar materials . And grouping of components that should be removed to comply with legislation (depollution) or cause problems for automated recycling process.
- Reuse and remanufacturing. Components suitable to reuse or remanufacture should be grouped into modules.

Fig. 6. Modularity drivers linked to product Lifecycle phases

3.3. Step 3: Formation of modular product architectures

In this step of the framework, a multi-objective optimisation algorithm is applied to generate a set of different modular product architectures. As there is no existing algorithm suitable for this task a custom multi-objective grouping genetic algorithm (MOGGA) has been developed [42]. In the software prototype, the MOGGA is applied to find a whole set of optimal modular architectures through manipulation of the component interface and lifecycle similarity data in the various matrices. Each solution is found by varying the membership of components to modules, in the interaction matrix, such that the developed modularity metrics (Eq. (1) and (2)) are maximised for seven different objectives (coupling, design, purchase, assembly, sales, use and end-of-life). Of course it will often be impossible to simultaneously maximise every objective, so different trade-off solutions are produced. Thus in a single optimisation run the MOGGA is able to produce a whole set of solutions that represent a good coverage of the trade-off surface.



		Components:	Air Controls	Refrigeration Controls	Sensors	Command Distribution	Blower Controller	Heater Hoses	Radiator Engine	fan Condenser	Compressor	Accumulator	Evaporator Core	Heater Core	Blower Motor	Evaporator Case	Actuators	
Coupling Interaction Matrix		Air Controls	1		0.4	0.4	0.1					0.1						0.1
		Refrigeration Controls		1		0.3						0.1	0.3					
		Sensors	0.4		1	0.3												
		Command Distribution	0.4	0.3	0.3	1	0.3			0.3		0.3				0.3		0.3
		Blower Controller	0.1							0.3	1	0.7	0.2					
		Heater Hoses						1							0.3			
		Radiator							1	0.7	0.7							
		Engine fan				0.3			0.7	1	0.7							
		Condenser							0.7	0.7	1	0.3		0.5				
		Compressor	0.1	0.1		0.3					0.3	1	0.4	0.3				
		Accumulator		0.3								0.4	1	0.4				
		Evaporator Core									0.5	0.3	0.4	1		0.2	0.5	
		Heater Core						0.3							1		0.5	
		Blower Motor				0.3	0.7							0.2		1	0.4	
		Evaporator Case					0.2							0.5	0.5	0.4	1	0.2
		Actuators	0.1			0.3											0.2	1
lifecycle driver Mapping	Design	Control team	1	1	1	1	1											
	Design	Air flow team														1	1	1
	Design	Heating system						1	1	1					1			
	Design	air con system team									1	1	1	1				
	Purchase	supplier 1	1	1	1	1	1											
	Purchase	supplier 2										1						
	Purchase	Make in-house						1	1	1	1		1	1	1	1	1	1
	Assembly	Passenger side	1	1	1	1	1						1	1	1	1	1	1
	Assembly	Engine side (radiator)						1	1	1	1							
	Assembly	Engine side (other)						1				1	1					
	Sales	Hose length						1										
	Sales	Command distribution Length				1												
	Sales	Air flow pattern															1	1
	Sales	manual/ auto models	1	1	1	1	1										1	1
	Sales	Radiator/ condenser size							1	1	1							
	End-of-life	Reuse								1		1				1		
	End-of-life	recycle (Shredder)						1	1		1		1	1	1		1	1
	End-of-life	Recycle (Remove for separate)	1	1	1	1	1			1						1		1

Fig. 7. Interaction matrix for car climate example

3.3.1. Modularity Metrics

Two modularity metrics have been developed for the framework; these are the module independence ratio and the module coherence ratio. These metrics are based upon two key modularity principles discussed in the literature: module independence and module coherence. The objective of modularisation from the module independence perspective is to achieve loosely coupled, independent modules. This can be achieved by ensuring that component interactions between modules are minimised. Module coherence is concerned with ensuring that components within modules are addressing the same modularisation objective. In the software implementation of the framework, the module independence metric is used for analysis of component coupling whereas the module coherence metric is associated with the component lifecycle similarity. Ideally, an optimal modular architecture will have loosely coupled independent modules that are highly coherent in terms of component lifecycle similarity. However, in reality, the two modularity concepts are contradictory. Improving the independence of modules will often mean that the coherence of modules deteriorates and vice versa. Furthermore, there will also be trade-offs needed between the different lifecycle phases. For example, a higher coherence in terms of the design stage may mean a lower coherence at the end-of-life stage. Hence an important part of the framework is to generate and evaluate a number of alternative modular architectures based upon the trade-offs between module independence and module coherence of different lifecycle objectives.

Module Independence (MI) ratio is the measure of component coupling within modules divided by the total coupling between all components and is given by Eq. 1. A higher ratio means more interactions are kept within modules rather than across modules, and the modules are more independent.

$$\mathrm{MI} = \sum_{\mathrm{m}-1}^{\mathrm{M}} \frac{\mathrm{CI_n}}{\mathrm{CI_{maxn}}} \qquad (1)$$

Where:

M = module number

$\mathrm{CI_n}$ = the number of couplings within module m

$\mathrm{CI_{maxn}}$ = the maximum strength of all component couplings

Module Coherence (MC) ratio measures the total number of components with the same modular driver needs divided by the maximum potential number of component similarity interactions within the modules and is given by Eq. 2.

$$\mathrm{MC} = \sum_{\mathrm{m}-1}^{\mathrm{M}} \frac{\mathrm{SI_n}}{\mathrm{SI_{maxn}}} \qquad (2)$$

Where:

M = module number

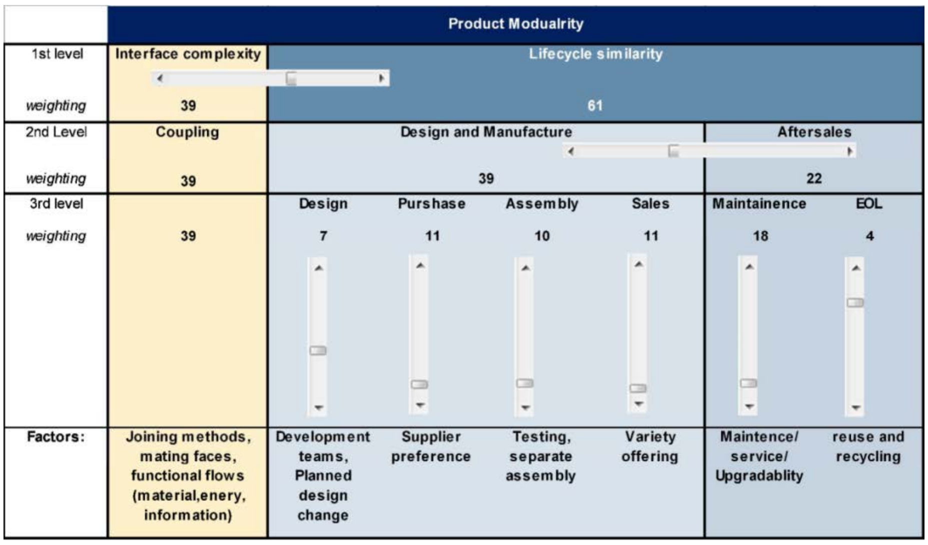
$\mathrm{SI_n}$ = total lifecycle similarity interactions within module m

$\mathrm{SI_{max\ n}}$ = total possible lifecycle similarity interactions within module

3.4. Step 4: Exploration of different modularisations scenarios

In this step of the framework multi-criteria decision making is used to support the solution ranking and scenario analysis stage. This enables the user to explore compromises and ultimately choose the most suitable modular structure from the different alternatives available. Because in the previous step a solution set has already been generated scenario analysis out can be carried out in ‘real time’.

In the software implementation, a product modularisation objective hierarchy has been developed based upon the analytical hierarchical process (AHP) used in multi-criteria decision making (Fig. 8). In AHP pair-wise comparisons of the each criterion are made at each level of the hierarchy, ultimately generating a weight vector for each objective. The weight vectors are then used to provide ranking of the solutions generated by the multi-objective algorithm. For example, starting at the top of the hierarchy the user may decide that component coupling is more important than lifecycle similarity and will set a greater weight by adjustment of the corresponding slider bar. The generated solutions will thus be ranked according to these preference weights and the highest ranking solutions presented to the user for further analysis. In the software prototype the modular solutions are represented by radar plots (Fig. 9). These plots give the user a suitable means for making comparisons between different solutions and exploring different modularisation scenarios.



	Product Modualrity						
1st level	Interface complexity	Lifecycle similarity					
weighting	39	61					
2nd Level	Coupling	Design and Manufacture				Aftersales	
weighting	39	39				22	
3rd level		Design	Purshase	Assembly	Sales	Maintainence	EOL
weighting	39	7	11	10	11	18	4
Factors:	Joining methods, mating faces, functional flows (material,enery, information)	Development teams, Planned design change	Supplier preference	Testing, separate assembly	Variety offering	Maintence/ service/ Upgradablity	reuse and recycling

Fig. 8. Modularisation lifecycle optimisation objective hierarchy

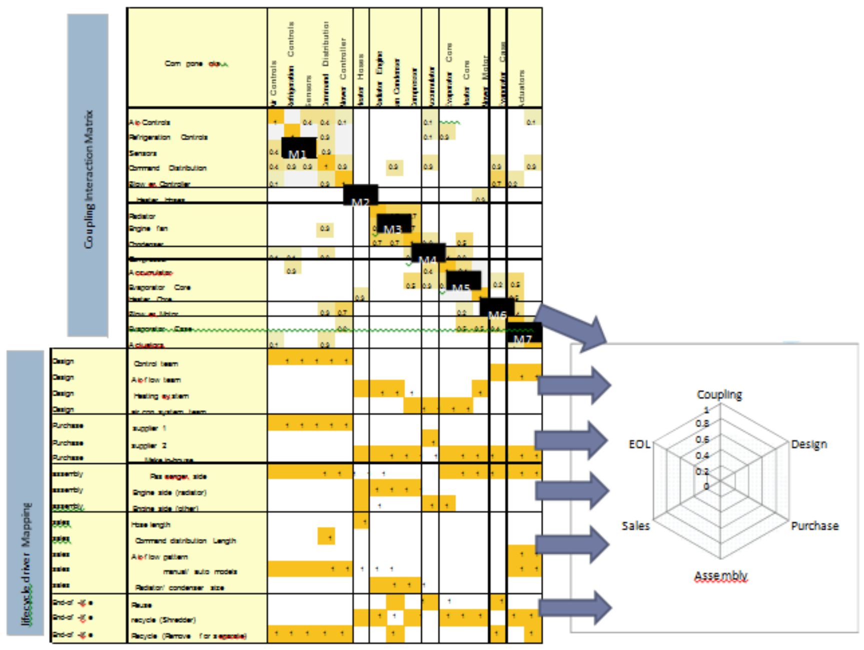
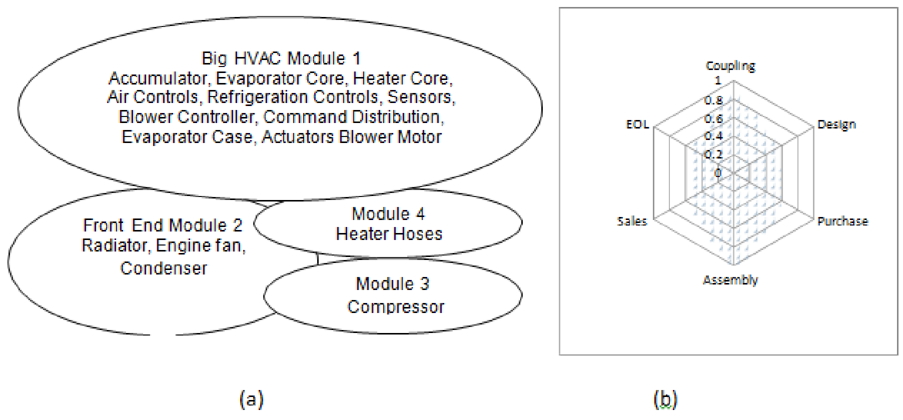


Fig. 9. Interaction matric showing recommended modularisation of car climate example

3.5. Chosen modular solution

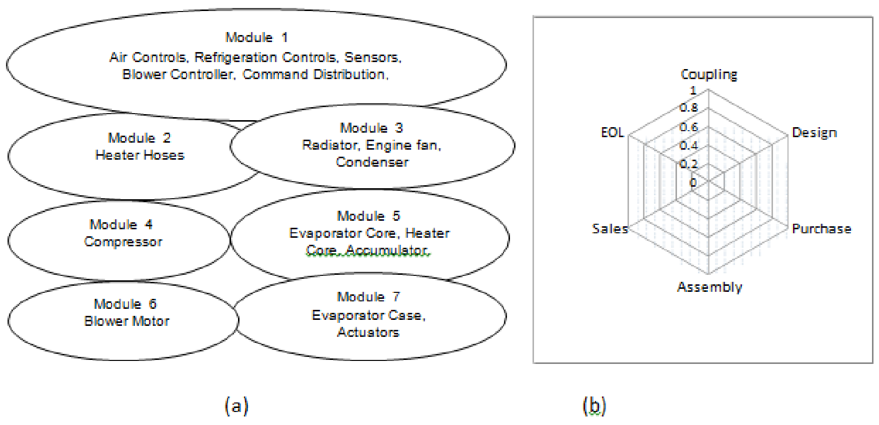
The chosen modular solution for the climate control system is shown in matrix representation (Fig. 8) and in pictorial form (Fig. 11). Although this is a hypothetical decision, it is a solution that offers good lifecycle performance. Furthermore, it is argued that the relatively poor performance of 'component coupling' can be tackled by the careful design of the interfaces between modules.

The chosen modular product architecture has been compared with an existing modular structure in a currently manufactured climate control system, (Fig. 10). This information comes from [37], who performed the case study with a tier one automotive supplier. Nepal discusses that the current climate control system was not systematically modularised in the past, and hence very few modules existed. The existing architecture may well be optimal for assembly time, however, when evaluated using the developed framework and metrics it is poorly configured for the other lifecycle considerations. It is argued however that the grouping of components with similar lifecycle characteristics using the developed framework will significantly reduce associated lifecycle costs. This will include module replacement costs, the costs of implementing variety, cost saving from module sharing (common product platforms) and improvement of recycling and reuse revenues. However, for verification, it is recommended that a detailed cost analysis should be done after a suitable modular structure has been found. Detailed cost modelling of product modularity was considered out of the scope of current research presented in this paper, but does offer an interesting area of future work.



Big HVAC Module 1
Accumulator, Evaporator Core, Heater Core, Air Controls, Refrigeration Controls, Sensors, Blower Controller, Command Distribution, Evaporator Case, Actuators Blower Motor
Front End Module 2
Radiator, Engine fan, Condenser
Module 4
Heater Hoses
Module 3
Compressor
Coupling
1
0.8
0.6
0.4
0.2
0
Design
Purchase
Assembly
Sales
EOL
(a)
(b)

Fig. 10. Existing modularisation of car climate control system a) according to Nepal, (2005) b) modularity analysis with framework



Module 1
Air Controls, Refrigeration Controls, Sensors, Blower Controller, Command Distribution,
Module 2
Heater Hoses
Module 3
Radiator, Engine fan, Condenser
Module 4
Compressor
Module 5
Evaporator Core, Heater Core, Accumulator,
Module 6
Blower Motor
Module 7
Evaporator Case, Actuators
Coupling
1
0.8
0.6
0.4
0.2
0
Design
Purchase
Assembly
Sales
EOL
(a)
(b)

Fig. 11. Recommended modularisation of car climate control system a) modules and components b) modularity analysis with framework

4. Conclusions and Future work

In this paper product modularity has been defined as a complex configuration problem in which the product system is decomposed into smaller more manageable chunks (modules) to improve product lifecycle performance. However, the majority of previous modularity methods have used simplistic optimisation models to handle an inherently complex multi-objective problem. The framework presented in this paper provides an alternative method of product modularisation for lifecycle optimisation using a novel multi-objective approach. This is primarily based upon application of a multi-objective algorithm to generate a whole set of alternative modular architectures, followed by multi-criteria decision making to help identify the best compromise solution according to the company's strategic goals. The focus of this paper has been to describe the framework's four main steps using an example product: the car climate control system. From this study, it is clear that this product is not optimised for the whole product lifecycle and that alternative modular architectures should be explored to improve the product's overall lifecycle performance.

In the research presented in this paper an assumption has been made that one overall modular structure will be chosen for the product. That is a modular architecture that attempts to optimise the whole product lifecycle. However, for some products it may be more sensible to create multiple modular structures within the product; each optimised for different lifecycle phases. This may of course add further complexity to a modular product strategy; and would require the definition of addition optimisation goals. Future work will thus expand the scope of the developed framework to explore the possibility of lifecycle optimisation with multiple modular structures within the same product.

5. Acknowledgements

This work was funded through a research scholarship awarded by the School of Mechanical and Manufacturing Engineering at Loughborough University, and this is gratefully acknowledged.

References

1. Krogstie, L., Ebro, M. and Howard, T. J. (2014). How to implement and apply robust design: insights from industrial practice. *Total Quality Management & Business Excellence*, published online July 2014, 1-19.
2. Ulrich, K., and Tung, K. (1991). Fundamentals of Product Modularity. In *Issues in Design/ Manufacture Integration 1991*, edited by A. Sharon (New York: ASME), DE 39.
3. Ulrich, K. (1995). The Role of Product Architecture in the Manufacturing Firm. *Research Policy*, 24, 419-440.
4. Nepal, B. P., Monplaisir, L. and Singh, N. (2006). A Methodology for Integrating Design for Quality in Modular Product Design, *Journal of Engineering Design*, 17(5), 387-409.
5. Stone, R. B., Wood, K. L., and Crawford. R. H. (2000). A Heuristic Method for Identifying Modules for Product Architectures. *Design Studies*, 21(1), 5-31.
6. Baldwin, C.Y. and Clark, K.B. (2000). *Design Rules, Volume 1: The Power of Modularity*. Cambridge, MA: MIT Press.
7. Kusiak, A. and Huang, C-C, (1997). Design of Modular Digital Circuits for Testability, *IEEE Transactions on Components, Packaging and Manufacturing Technology*, Part C, 20(1), 48-57.
8. Sanchez, R, (1999). Modular Architectures in the Marketing Process, *Journal of Marketing*, 63, 92-111.
9. Suh, N.P. (1990). *The Principles of Design*, New York: Oxford University Press.
10. Pahl, G. and Beitz, W. (1984). *Engineering Design: A Systematic Approach*, Berlin: Springer-Verlag.
11. Jiao, J and Tseng, M.M. (2000). Fundamentals of Product Family Architecture, *Integrated Manufacturing Systems* 11(7): 469–483.

12. Agard, B. and Bassetto, S. (2013). Modular Design of Product Families for Quality and Cost. *International Journal of Production Research*, 51(6), 1648-1667.

13. Ericsson, A. and Erixon, G. (1999). *Controlling Design Variants: Modular Product Platforms*. New York, NY: ASME Press.

14. Newcomb, P.J., Bras, B., and Rosen, D.W. (1996). Implications of Modularity on Product Design for the Life Cycle. *Proceedings of the 1996 ASME Design Engineering Technical Conferences*, 8th International Conference on Design Theory and Methodology (Irvine, CA), 96-DETC/DTM-1516.

15. Gu, P. and Sosale, S. (1999). Product Modularization for Life Cycle Engineering. *Robotics and Computer Integrated Manufacturing* 15(5): 387-401

16. Gershenson, J.K., Prasad, G.J. and Allamneni, S. (1999). Modular Product Design: a Life-cycle View. *Journal of Integrated Design and Process Science* 3(4): 13–26.

17. Sanchez, R. (2000). Modular Architectures, Knowledge Assets and Organizational Learning: New Management Processes for Product Creation, *International Journal of Technology Management*, 19(6), 610–629.

18. Mikkola, J.H. and Gassman, O. (2003). Managing Modularity of Product Architectures: Toward an Integrated Theory, *IEEE Transactions on Engineering Management*, 50(2), 204–218.

19. Qiao, H., Mo, R., Xiang, Y., Franke, M., Hribernik, K. A. and Thoben, K. D. (2014). An autopoietic approach for building modular design system. In: *2014 International ICE Conference on Engineering, Technology and Innovation (ICE)*, 1-7, IEEE.

20. Issa, H., Ostrosi, E., Lenczner, M. and Habib, R. (2013). Influence of Functional Knowledge Structuring for Modular Design. *Advanced Materials Research*, 651, 595-600.

21. Haiqiang, L., Guihu, C. and Lv, M. I. N. G. (2014). A Control Algorithm Based on Modular Design Approach to Complex Product Engineering. *Sensors & Transducers* 169, 1726-5479.

22. Gershenson, J.K., and Prasad, G.J. (1997b). Product Modularity and its Effect on Service and Maintenance. Proceedings of the 1997 Maintenance and Reliability Conference (MARCON), Knoxville, TN.

23. Adomeit, A., Lakshmanan, M., Schervan, T., Dafnis, A. and Reimerdes, H. G. (2013). Structural Concept and Design for Modular and Serviceable Spacecraft Systems. Proceedings of 54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials conference.

24. Gershenson, J.K., and Prasad, G.J. (1997a). Modularity in Product Design for Manufacturing. *International Journal of Agile Manufacturing* 1(1): 99–109.

25. Song, J. Y., Wu, L. K. and Sheng, Z. Q. (2013). Development of Modular Design System for CNC Machine Tools. In: *Proceedings of 20th International Conference on Industrial Engineering and Engineering Management*, 69-77, Springer Berlin Heidelberg.

26. Zhang, Y. and Gershenson, J.K. (2002). Questioning the Direct Relationship between Product Modularity and Retirement Cost, *Journal of Sustainable Product Design*, 2, 53-68.

27. Guo, F. and Gershenson, J.K. (2007). Discovering Relationships between Modularity and Cost, *Journal of Intelligent Manufacturing* 18(1): 143-157.

28. Liu, H. Y. (2014). Research of Modular Design in Disassembly. *Applied Mechanics and Materials*, 552 70-75.

29. Ji, Y., Jiao, R. J., Chen, L. and Wu, C. (2013). Green modular design for material efficiency: a leader–follower joint optimization model. *Journal of Cleaner Production*, 41, 187-201.

30. Yang, J. R. and Liu, Q. Y. (2013). Design Method Research on Green Modular that Oriented Remanufacturing Engineering. *Applied Mechanics and Materials*, 365, 545-548.

31. Ishii, K. and Lee, B. (1996). Reverse Fishbone Diagram: A Tool in Aid of Design for Product Retirement. *Proceedings of the 1996 ASME Design Engineering Technical Conference*.

32. Blackenfelt, M. (2001). *Managing Complexity by Product Modularisation*, Doctoral Thesis, Royal Institute of Technology, Stockholm, Sweden.

33. Pimmler, T. U. and Eppinger, S. D. (1994). Integration Analysis of Product Decompositions, *Proceedings of ASME Design Engineering Theory and Methodology Conference*, DE Vol. 68.

34. Kreng, V.B. and Lee, T-P., L. (2004). Modular Product Design with Grouping Genetic Algorithm - a Case Study, *Computers and Industrial Engineering*, 46(3), 443-460.

35. Whitfield, R.I., Smith J.S. and Duffy, A.H.B. (2002). Identifying Component Modules, *7th International Conference on Artificial Intelligence in Design (AID'02)*, Cambridge, UK.

36. Erixson, G. (1996). *Design for Modularity, In Design for X: Concurrent Engineering Imperative*, Chapman & Hall: New York, ed G.Q. Huang, 356-379.

37. Nepal, B. (2005). An Integrated Framework for Modular Product Architecture, Ph.D. Dissertation, ETD Collection for Wayne State University. Paper AAI3198695.

38. Saaty, T. L. (1990). How to Make a Decision: The Analytic Hierarchy Process, *European Journal of Operational Research*, 48(1), 9-26.

39. Stone, R.B. (1997). Towards a Theory of Modular Design, Ph.D. Dissertation, University of Texas at Austin, USA.

40. Andreasen, M.M. (1998). Reduction of Complexity of the Product Modelling by Modularization, *Proceedings of Produktmodeller-98*, Linköping, Sweden.

41. Behr. (2010). www.behr.de, accessed May 2013.

42. Lee, M.J. (2010). *Product Modularity: A Multi-objective Configuration Approach*. Ph.D. Thesis, Loughborough University, UK.