Implementing Secure Modular Design of Configurable Products, a Casestudy

Henk Jan Pels

Implementing Secure Modular Design of Configurable Products, a casestudy

Dr.ir. Henk Jan Pels,
Phi Knowledge Process Enabling b.v.
h.j.pels@outlook.com

Abstract.
Secure modular design is a method to prevent failures in complex configurable product. After introducing the theory behind the method this paper describes a case study in implementing this method in a real mechanical engineering environment. Much attention has been paid to involving engineers and management in setting up the implementation process, especially in translating the abstract mathematical concepts in the theory to concrete mechanical concepts. Although the method requires serious additional effort in recording design constraints, the participants in the case study expressed confidence in the value of the method.

Keywords: modular design, product configuration, module independence, change management, constraint management

1 Introduction

The market asks for ever increasing product variety to better meet individual customer needs (12). Configure to order (CtO) production is an answer to this trend. A configurable product offers options with choices, such that the customer can configure his product by filling in his choice per option. Based on these choices the product configuration system generates the production information and the customer specific product can be produced. Often the product has a modular design, where each module implements specific options and choices. Each module is a product family where each set of choices specifies a module variant. Changing a choice can be seen as exchanging one module variant for another.

When developing the product, each module variant must be verified and tested. When there are 10 modules with 10 variants each, this means 100 module variants to be verified and tested. What makes it worse is that there may be unanticipated relationships between specific variants of different modules that may cause product failure if the combination of those two variants is selected, even when there is no direct interaction between those module variants. This means that to be sure that non of the possible configurations will cause failure during production or operation, each of the 10**9 configurations should be verified individually, which clearly is not feasible. Because of this problem we see many failures of products during operation, caused by unforeseen interactions between modules. What makes the verification of modules in configurable products more complex is the problem of constraint propagation (13): constraints can form chains through the product: a constraint between parameters A and B and a constraint between B and C infers a constraint between A and C. In this way long chains

of relationships between design parameters may occur that reach from one far corner of the product to another.

The method for 'secure modular design of configurable products' as described in (1) proposes a theoretical solution for this problem. The method depends on formal rules for the specification of module interfaces depending on specified design constraints. The formal rules make the method rather abstract and difficult to introduce in engineering practice. Constraint management requires additional effort, which raises doubt on the effectivity of the method. This paper describes a case study of implementing the method in an engineering organization, where the focus is on translation of theoretical into practical concepts. For the theoretical description of the method we refer to (1).

The structure of the paper is as follows. Section 2 explains the environment of the case study and the expected problems. In section 3 we present a summary of the theory behind the method, while section 4 describes the case study. Sections 5 and 6 provide discussion and conclusion.

2 Research Question and Approach

In change management (8) it is known that every change in a product may have impact: intended impact in better properties of the product, as well as unintended impact in causing conflicts with elements of the product outside the changed part. Such conflicts may cause failure of the product during production or operation. Therefore, each change requires thorough impact analysis and serious verification and test before release. This implies that for configurable products each module variant must be verified and tested before it can be released for production, not in isolation, but in every possible configuration of the product. Breaking down the product into modules does not solve the problem because adding a module variant is not different from changing a part.

The Secure Modular Design Method, as described in (1), proposes a solution for the problem above by introducing the concept of module independence: an independent module needs only by verified against the specifications visible in its interface. A short description of the method is given in section 3.2. Since the method is derived from information systems theory and based upon formal specification of design constraints, its application requires good understanding of some abstract concepts. Since most mechanical engineers are not trained in using such concepts there is a serious risk that it will take a lot of effort to implement the method in a mechanical engineering environment. This leads to the research question:

is it possible to implement the rather abstract secure modular design method in a very practically oriented engineering process?

A Dutch company, designing and manufacturing trailers, was interested in implementing modular design. This company produces trailers in a CtO and EtO approach. To be more competitive they wanted to expand on product variety in the configure to order market and to reduce on ever growing engineering cost. In order to reduce production cost, this company had moved the production of their trailers to a factory in

Bosnia. In order to reduce engineering cost, most order dependent engineering was moved to the Bosnian engineering department as well, leaving only new product development in the Netherlands.

Secure modular design seemed a promising approach, so they decided to implement the method. This enabled us to execute a use case in the company. Two main problems are foreseen in the implementation process:

1. Is it possible to explain the method in concepts that are close to the mechanical engineering profession?
2. The method requires elicitation, recording and management of all constraints in the product design. This is additional work and not immediately rewarding. Can a proper database tool help to reduce the effort to an acceptable level?

The approach of the case study is based upon principles to overcome resistance to change as proposed in (3): first resolve differences in professional discourse between the parties involved and second create interactions between all operational and management levels involved.

3 Theory of Secure Modular Design

Before describing the case a short introduction of the theory behind secure modular design will be given.

3.1 Modular Design

What makes the difference between a module and a part? The generic meaning of module is “exchangeable component” (4, 5). A basic requirement for exchangeability is compatibility of interfaces, so specification of interfaces is essential for modular design. The original aim of modular design is to enable the product to be adopted to changing needs of the user by exchanging modules (6, 7). For decomposing a system into modules often the Design Structure Matrix method (10) is used: components with relatively more interactions are combined into one module, in order to minimize interactions between modules and to simplify interfaces. Other research aims at automation of the decomposition process, based upon mathematical theory of combinatorial engineering (9). The importance of constraints in mechanical design is analyzed in (11). All these methods provide guidance in how a product can be broken down into modules. However, they do not address the problem of how to make sure that a new module variant will not cause failure in other modules. To our knowledge the use of design constraints to manage change impact as proposed in (1) is unique until now.

Following database theory (2) a constraint can be understood as a predicate over one or more design parameters of the product. A design satisfies the constraint when the predicate is true. Requirements like ‘payload < 1000kg’ are constraints that are input for the design. Design considerations like ‘diameter axle < diameter hole’ are constraints that follow from engineering design considerations. Product failure occurs when during production or operation a product state occurs that violates a constraint

e.g. when because of extreme temperature the axle fits gets stuck in the hole. The problem in complex products is that constraints can form chains via shared parameters, (constraint propagation) such that a change of state in one part can cause a conflict in another part, even if both parts have no direct relationships. The problem of constraint propagation is also discussed in the context of design compilers (13).

The purpose of the method described in (1) is to define a module in such a way that it can be designed and verified as an independent unit, without knowledge, other than specified in its interface, of the design of the system in which it is integrated. If a module can be specified in such a way, there is no need to test it in each individual configuration in which it can be applied. As a result, the risk of product failure could be largely reduced.

Because of the essential role of interfaces, the method uses the following definition of module:

A module is a part with a defined interface.

The interface contains the product specifications that the module shares with its environment. This interface can be split into two domains:

1. the *foreign domain* that contains the specifications of the product outside the part, that must be known to the designer of the part,
2. the *public domain* containing the specifications of the part that must be known by designers of other parts in the product.

In this definition the part is the artifact as specified in the CAD model. The set of its specifications is called the *own domain*. A good module has a small interface, meaning that the designer of the module only needs limited knowledge of the target product, making the module less sensitive to changes in its environment, while it shows only limited detail to its environment, giving the designer more freedom in changing his module without external impact.

3.2 Secure Modular Design

When changing a design parameter of a module, the designer must be able to check, all constraints that can possibly be violated by this change. Constraint propagation complicates this task. For secure design we need modules that:

can be verified and tested to operate secure in the intended product, without knowledge of that product other than specified in the module's foreign domain.

Such a module is called *independent in the product*. This concept of module independence has originally been developed for data-based information systems (2). Such systems consist of a set of applications that operate on a common database. Database constraints limit the allowed values of the data elements in the database. Because databases and their constraints can be specified in formal mathematical terms, database properties can be proven in a formal way. In this way it has been proven (2) that:

a module is independent in a database system, if every constraint in the database schema that refers to an element of the modules own domain, does not refer to any element outside the module.

From this condition it follows that module independence depends on the specification of its interface in relation to the constraints that must hold for the product as a whole. For an independent module constraint chaines beyond its foreign domain can be ignored. Note that independence is not a property of the module on its own, but a property of the module in a specific product configuration. A module that is independent in one configuration may not be independent in another configuration. Replacing one module in a system may cause loss of independence of other modules.

Proving independence of a module depends on the constraints in the system. This means that for secure design all constraints must be explicitly known. Integration of a new module in a system implies:

1. Proving independence by checking all constraints that refer to the module,
2. Verifying the module against all constraints that refer to its own domain.

After these steps the module cannot cause any conflict with a constraint anywhere in the system. Note that in most modular design methods interfaces are viewed as gates for transfer of space, energy, information or material (1). In our approach the interface is the windows through which the module and the system view each others specification.

If a mechanical product is specified in a 3D CAD system, each part is specified in terms of features with parameters (e.g. a cylinder with length and radius). In that sense a CAD model is like a database where each feature is an object in the database and each choice of parameter values a state of that object. Designing a part is then defining the proper features and choosing optimal values for their parameters. In this way the concept of module independence can be applied to mechanical systems and proven on basis of the condition for independence as formulated above.

Applying these steps in designing a modular product is the essence of "secure modular design". The main concepts used in the method are illustrated in the object diagram in figure 1. In the notation used each block is an object class with its attributes. Each line is a relationship between object classes that allows expressions over multiple objects. Unless explicitly labeled the link gets the name of the opposite class: if f is a Feature then f.Module is the module to which the feature belongs. An arrow at one end means a N:1 relationship. No arrows means an N:M relationship: f.Parameter is the set of parameters of Feature f.

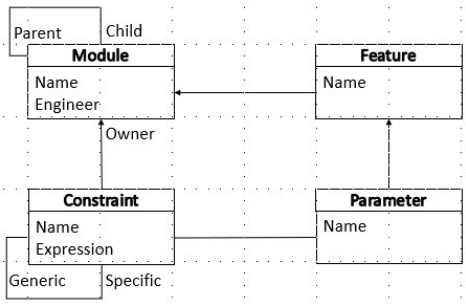

Fig. 1. Object diagram for secure modular design.

Figure 1 tells us that modules are structured in Parent-Child relationships: if m is a module then m.Parent contains the higher level modules of which m is a part. Note that a constraint can refer to many parameters and that the same parameter may occur in several constraints. There are constraints that are not specific for one module or even one product, but yet are relevant for the design. These are for instance legal rules, production rules, engineering rules, product requirements and customer order requirements. When designing his module, the designer will formulate module specific constraints to obey relevant generic rules. Therefore, a constraint can have a link to a generic constraint, meaning that the specific constraint is based upon that generic constraint. This construction also enables to trace where higher-level constraints are implemented in modules. Since requirements are constraints as well, this link provides an easy way to trace where changed requirements have impact. Generic constraints can be organized in generic modules that are not part of a product but serve as container for a group of constraints.

Also, constraints can be structured: if c1 and c2 are constraints and

C2. Generic = c1,

then c2 is based upon c1. This can be the case when c1 is a legal requirement and c2 is this requirement expressed in the parameters of the design.

From this structure for each module a list can be derived of all parameters that occur in constraints that refer to at least one parameter of the module. After filtering out the constraints that are local for the module, this list shows exactly the minimum foreign domain to make the module independent. In formal notation the foreign domain of module m can be expressed as:

$$\{p1 \in \text{Parameter}: \exists\, c \in p1.\text{Constraint}: \exists\, p2, p3 \in c.\text{Parameter}: p2.\text{Feature.Module} = m \wedge p3.\text{Feature.Module} \neq m\}.$$

Listing these parameters together with their module and features shows exactly the foreign domain of m, as required for module independence. If constraints are specified in formal language, then the tables Feature and Parameter can be derived by an algoritm.

4 Implementing Secure Modular Design: a Case Study

The method requires that all constraints in the product are specified explicitly. In mechanical engineering this is not common practice: only requirements are documented but design considerations are directly translated into design solutions without further documentation. Also, specification of mechanical constraints in formal language is not the core competence of most mechanical engineers. Further the condition for module independence seams simple in its abstract formulation but appears to be rather difficult to apply to a concrete design. This means that applying the method requires quite some intellectual exercise from the engineers which implies a large risk for non-acceptance of the method. Therefor the use case started with formulation of an implementation plan.

As mentioned in section 2 changing the design process requires careful analysis of differences in the professional discourse of parties that propose the change and parties

that are involved in the changed process. An implementation plan was conceived along the following steps:

1. Develop a Secure Modular Design course and take care for the professional language,
2. Present course to engineers and production management,
3. Initiate design project for a new modular configurable trailer,
4. Guide designers in applying the method,
5. Discuss, evaluate and improve method.

These steps will be discussed below.

4.1 Develop Secure Modular Design Course

The course has been developed in close cooperation between the researcher and the chief engineer. In the process it appeared that the formal mathematical concepts in which the method was defined, were difficult to explain in a mechanical context. Much effort was spent on translating mathematical concepts into mechanical terms. Also, the steps to introduce the abstract concept of module independence in mechanical terms, were carefully chosen. Since the essential part of the method is to identify and specify all relevant constraints, we found that a tool for managing constraints was needed. A prototype for such a tool was developed in Microsoft-Access. Where the description of the method in (1) focusses on formal and scientific explanation, the course should focus on ability to apply the method reliably in practice.

Before starting the course for the engineers, the course was presented to the company management. Since the method was new, also for the management, we found that the management should understand the method and be given opportunity to accept or reject the risk of engaging in such a novel approach. After the presentation, the management decided to fully support introduction of the method, which was felt to be important to motivate the organization to adopt it.

4.2 Present Course to Engineers and Production Management

The method was presented in Bosnia as a one-day course for engineering, production preparation, and quality management. Not only engineers, but also employees from other function related to production were invited in order to create the required common professional discourse. The participants were given exercises to identify and formulate constraints. From unstructured discussions during and after the course it appeared that the method was well accepted. Engineers from design as well as production stated that that errors due to neglected constraints often occurred. The quality control manager explicitly stated that he expected substantial reduction of quality cost from them method. Although these expressions of acceptance may partly have been motivated by politeness, they surely formed a basis for a positive attitude towards continuation of the project.

4.3 InitiateDdesign Project for New ModularCconfigurable Trailer

In order to apply the method in practice a design project was defined for a modular container carrier. A real design project was needed in order to give the engineers real budget to spend on the project. The container carrier (figure 2) was chosen for its relative simplicity.

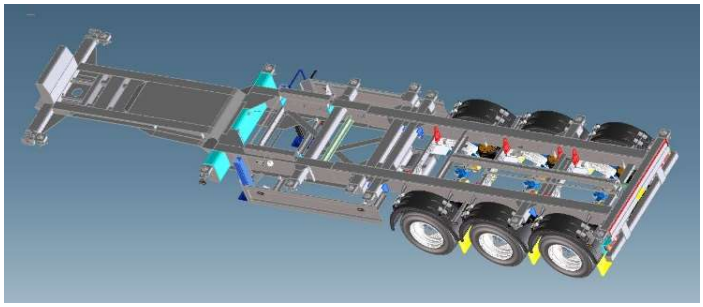

Fig. 2. Container carrier trailer

The steps for a modular design project were defined as:

1. Module architecture design,
2. Issue design order per module
3. Support engineers in design process and evaluate method,
4. Verify module designs,
5. Release trailer,
6. Evaluate method.

The architecture design resulted in modules as shown in Figure 3.

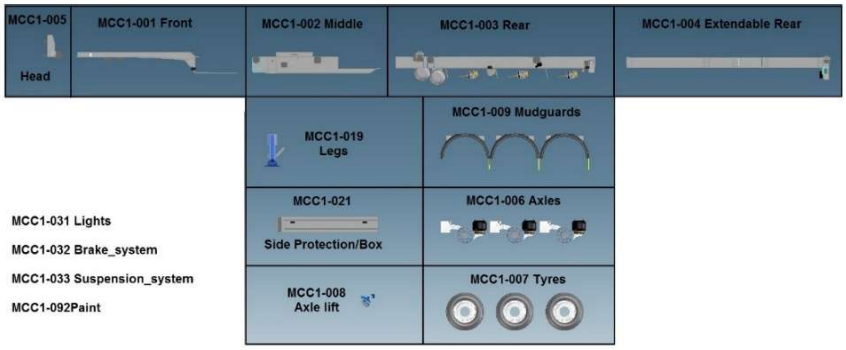

Fig. 3. Modular decomposition of trailer

This decomposition is based upon intuitive understanding of functional clusters. Also, constraints at product level, like commercial requirements en legal regulations were specified. Optimization for modularity criteria was postponed to a later phase.

4.4 Guide Designers in Applying the Method

A formal start of the project was made in a second one day course. Where the function of the first course was to introduce the concept of secure modular design and to create acceptance, the main goal of the second course was to introduce the engineers in using the constraint management tool. The tool, as presented in the first course, was largely conform the object model of figure 1. However, while developing the presentation for this second course we realized that the concepts used in the tool were still too distant from the current way of working of the engineers and thus required to many mental steps from the usual design task to specifying the constraints. Applying the formula at the end of section 3.1 is quite an intellectual challenge indeed! Therefore, the tool was reshaped and simplified. Figure 4 shows the Ms-Access database structure of the improved constraint management tool.

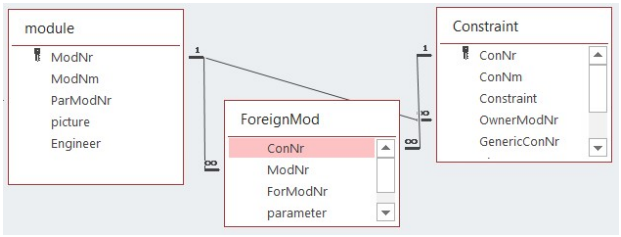

Fig. 4. Constraint management database structure

Modules and constraints are identified with a number. Although these numbers are not required in theory, it appeared that engineers are used to these numbers, so if the database offers no attribute for a number, they will integrate it in the name. The parent-child relationship is implemented in the ParModNr (parent module number) attribute. This implies that the same module can have only one parent module, but for the use case it was an acceptable simplification. Each module gets assigned a responsible engineer. Filling the module table is part of the architecture design.

Each engineer will have to specify the constraints owned by his module. Therefore, constraints can be entered per module using a subdatasheet. For this use case we allowed to formulate constraints in natural language.

To make a module independent, its foreign domain must contain all parameters that are referenced in in any constraint that also references a prameter in the own domain of the module. Therefore, it is essential to know for each constraint to what modules it refers. This must be specified by the designer, using table ForeignMod. The object model (figure 1) requires that each parameter is assigned to a feature, but in practice most features will be identified only later in the design process. Therefore, the concept Feature has been left out of the tool. For each constraint that refers outside his module,

the engineer must specify a ForeugnMod record in a subdatasheet. This specifies the other module and the referenced parameters. From the ForeignMod table the foreign domain per module can be derived in a query. Note that also constraints of other modules may induce foreign domain elements. These parameters form the foreign domain of the module and thus specify the interface that makes the module independent. Figure 5 shows the ForeignDomain query for some modules. Per module it shows the foreign module, the relevant constraints and the relevant parameters. This exactly shows for each module the foreign domain that makes it independent: the constraints that must be checked when verifying the module.

ForeignDomainQ

ModNr	ModNm	ForModNr	ForModNm	ConNr	ConNm	parameter	OwnerModN
MCC1-005	Head	MCC1-001	Front	C0009	Welding interface	Wgap5-1	Head
MCC1-005	Head	MCC1-001	Front	C0039	Module length	ml	Front
MCC1-005	Head	MCC1-002	Middle	C0005	CTC length twistlocks	head.twistlockle	Front
MCC1-005	Head	MCC1-003	Rear	C0005	CTC length twistlocks	head.twistlockle	Front
MCC1-005	Head	MCC1-004	Extendable Rea	C0005	CTC length twistlocks	CTCL	Front
MCC1-005	Head	MCC1-031	Lighting system	C0043	Pattern for sockets	sockettype	Head
MCC1-005	Head	MCC1-032	Brake system	C0042	Pattern for couplings	sockettype	Head
MCC1-006	Axles	MCC1-004	Extendable Rea	C0020	Height bumper	Angle_b	Extendable Rea
MCC1-007	Tyres	MCC1-004	Extendable Rea	C0020	Height bumper	Angle_b	Extendable Rea
MCC1-007	Tyres	MCC1-009	Mudguards	C0049	Radius mudguard	tyre.radius	Mudguards
MCC1-007	Tyres	MCC1-009	Mudguards	C0050	Width mudguard	tyre.width	Mudguards

Figure 5. view on ForeignDomain Query

4.5 Discuss, Evaluate and Improve Method

Because of the Corona crisis the use case temporarily got less priority. Thus, the design of the trailer could not yet be finished and evaluated.

5 Discussion

The main conclusion from this use case is that it is not impossible to convince an engineering organization of the potential of the Secure Modular Design Method. The second result is a prototype for a constraint management tool. Development of this tool learnt that the concepts of the theoretical background must and can be translated into a simplified structure as shown in figure 4.

Our expectation is that eliciting the constraints that touch a module is a good start for the module design process. Because constraints between two (or more) modules will probably be formulated by both engineers, the elicitation process will evoque effective discussion between engineers and ease the design process afterwards. Also, the tool will immediately show when a new of modified constraint touches other modules. This the responsiuble designers can be alerted to verify the new constraint.

Since the trailer design process is not finished, there are no trailers in production or operation, so it is not yet known whether the method actually reduces failure cost. However, the reactions of the staff on the introduction course showed that ignorance of constraints is a frequent source of failure, which indicates that the method has real potential to reduce failure cost.

The Generic-Specific relationship between constraints came up as a side effect of the use case. It was realized that recording all legal rules in the constraint database will be a huge effort which, however, can be spread over time by recording just the additional relevant rules per project. Since requirements are constraints as well, recording all constraint in the database will also be supportive to constraint management. The tool can show for each design detail on what constraints it is based and for each requirement it can be seen what design details in what products depend on this constraint. The constraint management tool will need integration with a requirement management tool, if in place.

In the tool as developed for the use case many details have been omitted that will be required for general use in a company. For instance, concepts for product, product family, customer order, design project will be needed. Also, constraint versioning related to module versioning will be needed. This means that integration with the PLM system will be required.

The method is based on the formal proof that an independent module that has been verified to obey all constraints in its interface, will not cause failure. However, this supposes that all relevant constraints have been identified and specified. In an information system, a constraint only exists if it is specified. In a mechanical system, constraints may follow from e.g. material properties and can be identified only as far as these properties are known. Therefore, there is always a possibility that a system with independent and properly verified modules will fail, because of an unforeseen constraint. Nevertheless, the method can strongly reduce the number of failures because in current practice constraints are used in design considerations, but not documented. Then in later changes the constraint is overlooked until it causes failure.

6 Conclusion

Although the Secure Modular Design Method is founded on abstract concepts that are quite distant from the mechanical design way of thinking, the use case shows that it is possible to make it accepted in an engineering department. By adapting the constraint management tool to the current wat of working, the additional effort for constraint management can be made feasible. The results so far are promising for better control of change impact in complex configurable products.

Acknowledgement

Execution of this use case was made possible in the context of the Dutch Government funded Private Sector Investment (PSI) Program 2014, project ‘High End Engineering & Manufacturing Technology Bosnia’, Group Jumbo Groenewegen & Eurometali. We also thank Group Jumbo Groenewegen, and the staff of the Hemtech Bosnia factory for their confidence and support in introducing the method in their organization.

References

1. Pels H.J. (2017) Secure Modular Design of Configurable Products. In: Ríos J., Bernard A., Bouras A., Foufou S. (eds) Product Lifecycle Management and the Industry of the Future. PLM 2017. IFIP Advances in Information and Communication Technology, vol 517. Springer, Cham.

2. Pels H.J. Geïntegreerde Informatiebanken, Modulair ontwerp van het conceptuele schema, Dissertation Eindhoven University of Technology, Stenfert-Kroese, Leiden, 1988 (in Dutch).

3. Pieterse, J., Service engineers in change: count your words, a case study into professional discourse and culture within three Dutch organizations, dissertation Open Universiteit Nederland, march 14 2014.

4. Huang, C. C. and Kusiak, A., Modularity in design of products. IEEE Transactions on Systems, Man, and Cybernetics, A, 28(1), 6677, 1998.

5. Gershenson, J. K., Prasad, G. J. & Zhang, Y., Product modularity: Definitions and benefits, Journal of Engineering Design, Volume 30 (2010) issue 3. 295-313.

6. Kusiak, A., Integrated product and process design: a modularity perspective, J. Eng. Design 2002, Vol13, No. 3, 223-231.

7. Pels H.J. "Modularity in Product Design", in Tichem M.e.a. (ed.) Proceedings of the 3rd WDK Workshop on Product Structuring, jun 26027, Delft University of Technolgy, 1998, ISBN 90-370-0169-6

8. Subrahmanian, E., Lee, C. & Granger, H. Managing and supporting product life cycle through engineering change management for a complex product. *Res Eng Design* **26,** 189–217 (2015). https://doi.org/10.1007/s00163-015-0192-1.

9. Levin, M. Sh., Modular systems design and evaluation, Springer, 1995.

10. AlGeddawy, ElMaragry, Reactive design methodology for product family platforms, modularity and parts integration, CIRP journal of Manufacturing Science and Technology 6 (2013) 34-43.

11. Scudieri, Paul; Lilly, Blaine, A Constraint-based Model of the Design Process in: Kovacevic, Ahmed, Ion, William, McMahon, Chris, Buck, Lyndon and Hogarth, Peter, DS 69: Proceedings of E&PDE 2011, the 13th International Conference on Engineering and Product Design Education, London, UK, 08.-09.09.2011 pp 205-210.

12. Erens, F.J., Verhulst, K. Architectures for product families, Computers in Industry**,** Volume 33, Issues 2–3, September 1997, Pages 165-178.

13. Ward, Allen C.: A Theory of Quantitative Inference of Artifact sets, Applied to a Mechanical Design Compiler. Technical report 1089 MIT Artificial Intelligence Laboratory, 1989.

14. Thomas U. Pimmler and Steven D. Eppinger, INTEGRATION ANALYSIS OF PRODUCT DECOMPOSITIONS, ASME Design Theory and Methodology Conference Minneapolis, MN September 1994, 343-351.