

Platform identification using Design Structure Matrices

Konstantinos Kalligeros*, Olivier de Weck, Richard de Neufville
Massachusetts Institute of Technology, Cambridge MA 02139

and Adrian Luckins
BP Exploration & Production, Sunbury, UK

Sixteenth Annual International Symposium of the
International Council On Systems Engineering (INCOSE)
8 - 14 July 2006

Copyright © 2006 Konstantinos Kalligeros. Published and used by INCOSE with permission.

Abstract: This paper introduces a methodology and algorithm for the qualitative identification of platform components at multiple levels of system aggregation, among variants within a family of systems. We assume that the architectural concept and the functional requirements for the variants are pre-determined, and use Sensitivity Design Structure Matrices (SDSM) to represent the sensitivities between the design variables of the variants. We then introduce a novel algorithm for the identification of platform variables given the SDSM for each variant. Finally, the methodology is extended to the qualitative identification of platforms at various levels of system aggregation, i.e., between systems, subsystems and components. The process is demonstrated in an automotive vehicle example of platform identification.

Keywords: Design Structure Matrix (DSM), platform, standardization, Design Rules

Introduction

Increasingly, a big contributor to competitive advantage is a firm's ability to balance between requirements for highly customized products and systems and standardized platforms. The motivation for customization comes from necessary change in product portfolios, embracement of new technology and evolution of product lines to changing consumer requirements. At the same time, production costs must be driven down using economies of scale, lead times must be shortened, and inventory must be minimized. Besides static goals such as the above, firms also strive to minimize risks (Ulrich 1995) and increase flexibility (Suh 2005) to respond to environmental threats and opportunities. Fricke and Schultz (2005) cite many examples from the automotive and other industries where product variety has increased over the past 20 years, while production costs have declined. This is the result of a relatively new and evolving body of literature, emanating both from the industry and academia, that advances and integrates concepts and methodologies, traditionally "owned" by disciplinary fields such as systems optimization or management science. Fricke and Schultz provide an excellent account under the name "Design for Changeability." A subset of these methodologies provides a solution approach to address the

* Corresponding author. Email kkall@alum.mit.edu

trade-off between customization and differentiation across multiple product lines, and the minimization of fixed and variable costs. An inclusive name for these methods is *product platforming*.

A *product platform* is a common set of subsystems, components, processes, interfaces etc. shared by all *variants* in a product family (Meyer & Lehnerd 1997). Platforms may emerge as product families evolve; in this case, the platform components and processes are those that are simply found to be common between variants. Alternatively, platforms may be imposed as a conscious decision on a collection of variants. In either case, platform components, systems or processes end up constraining the variants' design: because of the requirement that a platform component or process is identical in all variants, customized components are necessarily designed to be compatible with the platform; indeed, the range of possible (or desirable) customization in the entire product family is dictated by the platform (de Weck 2006).

For some systems, the choice of the platform systems or processes between variants is pretty simple: it may be intuitive or emerge naturally from the historical evolution of multiple variants. Potential platforms are those systems that act as "buses" in some way (Yu et al. 2003), or those that provide interfaces between other, customized systems. On the other hand, the deliberate identification of platforms is more difficult in network-like systems or systems in which platforms are comprised of subsystems and components from various levels of system aggregation. Platform identification is equally cumbersome in very large and complex systems.

This paper introduces a methodology for the qualitative identification of collections of subsystems and components that comprise feasible platforms. This contribution is relevant for systems in which platform identification is not intuitive or historically emergent. It is assumed that the architectural concept and the specifications for the product variants are given. The methodology is based on Design Structure Matrices (DSM, Steward 1981, 1991) and specifically the Sensitivity-DSM (Yassine and Falkenburg, 1999) to model how exogenous effects propagate through the interdependent components of a system. Related concept was presented in a seminal work by Sullivan et al. (2001). The main concept in this work is that the elements of the DSM that are not sensitive, directly or indirectly and within a certain tolerance, to exogenous changes, are potentially members of the platform collection of components. In short, the platform provides the *Design Rules* for the product family (Baldwin & Clark 2000).

The paper develops the concepts by first identifying platforms as collections of design variables using a novel algorithm. We then take a more qualitative approach to describe how the methodology can be used to identify platforms on multiple levels of system aggregation. The methodology is demonstrated using an example from the automotive industry.

Platforms as collections of design variables

All variants in a product family will share some commonality in the arrangement of components, their interactions, and their mapping between function and form (Martin and Ishii 2002). Variants in a product family thus share a *platform architecture*, i.e., a common "scheme by which the function of a product is allocated to physical components" (Ulrich 1995). This common architecture will most often be reflected in an identical system model between variants of a product line, i.e., an identical set of equations and variables that describe the system's response. Given a common set of variables that describe the family architecture for each variant, it is possible to represent the architecture of all variants within a product line in a Design Structure Matrix (DSM).

System Representation

DSMs provide a structured methodology for representing systems and processes. The term DSM summarizes a variety of different uses of essentially the same structure, i.e., a square matrix where each row (and the respective column) corresponds to a single “element.” Interactions between elements are represented as “1” (or another mark, often “x”) in the off-diagonal entries of the matrix body. Depending on what the elements represent, DSMs are usually referred to as “component-based,” “variable-based,”¹ “activity-based” or “team-based.” Interactions between components represent material or energy flows or even spatial relationships in a static system. A symmetric interaction between two variables means that they are coupled and need to be determined jointly; such interactions do not include any notion of time. Finally, interactions in activity-based DSMs represent precedence between tasks, therefore the order of the activities corresponds to the order in which activities are performed (Browning 2001).

A mapping usually exists between component, activity and team-based DSMs (Eppinger and Salminen 2001): system components are regarded as distinct line items in a work breakdown structure and are therefore treated as separate design activities. In turn, these design activities are assigned to separate teams. Furthermore, the relationship between a component-based and a parameter-based DSM is usually at the level of aggregation in describing a system. Consider for example a DSM representing the design of a system where each component is fully characterized by a single parameter: the component-DSM and parameter-DSM for such a representation would coincide. Establishing these links between the different DSM types will prove explanatory for the later part of the paper; this section focuses on parameter-based DSMs only.

Consider a system whose response and performance can be fully described using n variables $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, which are coupled in a model of equations. The corresponding variable-based DSM is the square matrix with n rows and columns, whose entries i, j and j, i are equal to “1” (symbolically, $DSM(i, j) = DSM(j, i) = 1$) if the two variables i and j are coupled. In this sense, a variable-based DSM is an N^2 matrix, and represents the architecture of a system. Particular values of the variables correspond to variants $\mathbf{x}^* = \{x_1^*, x_2^*, \dots, x_n^*\}$ within the architecture described by the DSM.

A sensitivity DSM (SDSM) is also a square matrix with n rows and columns. The entry i, j of an SDSM, however, represents the normalized sensitivity of parameter i to unit changes in parameter j in the neighborhood of the particular solution: $SDSM(i, j) = (\partial x_i^* / \partial x_j^*)(x_j^* / x_i^*)$. In other words, entry i, j represents the percent change in variable i caused by a percent change in variable j . For this reason, a sensitivity DSM is always more sparsely populated than a variable-DSM: a variable may depend on another variable, but its sensitivity to changes in the latter may be zero.

System or product variants exist to cover different commercial, marketing or societal needs and objectives that are completely exogenous to the system, i.e., design decisions cannot affect them. Examples of *exogenous factors* in the automotive industry are the condition of the roads in the region a vehicle is marketed and the typical weather. Marketing studies can translate exogenous factors to *functional requirements*. Functional requirements are performance or response targets the system has to meet, and as such they depend on both exogenous factors as

¹ A DSM whose elements represent scalar variables is usually referred to as a *parameter-based DSM*. The name *variable-based* is used here to avoid confusion with exogenous parameters, defined later.

well as the design variables of the system. Using the previous automotive example, a functional requirement affected both by the road quality as well as design variables is a measure of softness of a car's suspension system. Let the functional requirements for system variant \mathbf{x}^* be denoted by a vector $\mathbf{FR}^* = \{FR_1^*, FR_2^*, \dots, FR_m^*\}$.

The SDSM can be extended to include the vector of functional requirements (Figure 1). The south-western quadrant of the extended SDSM is populated by the sensitivities of design variables to exogenous parameters; the main body of the SDSM (south-east quadrant) contains the sensitivity of design variables to other design variables for the particular solution.

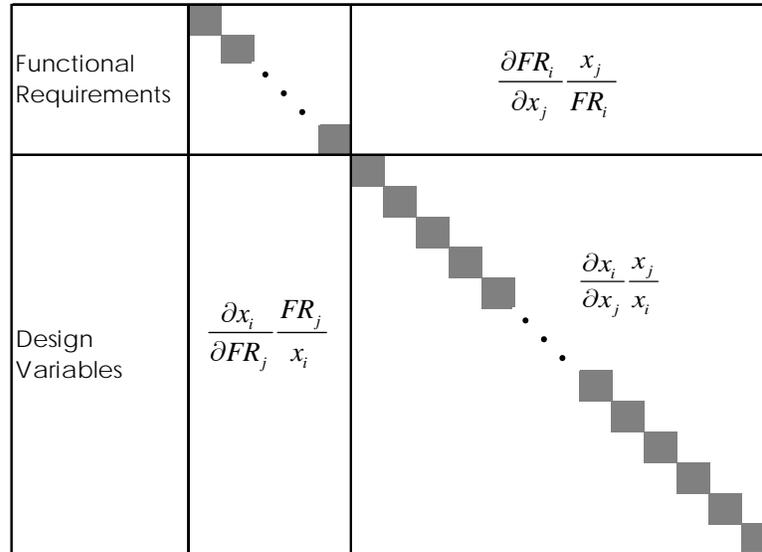


Figure 1: Normalized SDSM, extended to include exogenous parameters

Change propagation

Consider a particular solution $\mathbf{x}^* = \{x_1^*, x_2^*, \dots, x_n^*\}$ to the system model, and a small change $\Delta \mathbf{FR}$ in some of the functional requirements \mathbf{FR}^* on which this solution was based. The question is to find those design variables that will need to change to facilitate this perturbation in \mathbf{FR}^* , and those that may remain the same.

Assume the system model behaves linearly for the changes in design variables necessary to achieve a perturbation $\Delta \mathbf{FR}$ in the functional requirements. Then each design variable x_i will need to change by Δx_i :

$$\Delta x_i = \sum_{j=1}^m \frac{\partial x_i^*}{\partial FR_j^*} \Delta FR_j^* + \sum_{s=1}^n \frac{\partial x_i^*}{\partial x_s^*} \Delta x_s^* \quad (1)$$

This simply says that the required change in x_i is the cumulative change caused by all the functional requirements and other design variables to which x_i is sensitive in the neighborhood of x_i^* .

If every term in the sums in equation (1) is zero, then Δx_i will be zero. Writing this separately for each summation we obtain conditions (2) and (3). These conditions are sufficient but not necessary: in theory, the sum of the terms in equation (1) can be zero without necessarily

all the terms being zero.

$$\frac{\partial x_i}{\partial FR_j} \Delta FR_j = 0 \quad \text{for all } j = 1 \dots m \quad (2)$$

$$\frac{\partial x_i}{\partial x_s} \Delta x_s = 0 \quad \text{for all } s = 1 \dots n \quad (3)$$

Conditions (2) and (3) indicate whether the change introduced in the functional requirements propagates to design variable x_i . A change can propagate to variable x_i because x_i directly depends on an affected functional requirement, or because x_i is sensitive to changes in some other variable that in turn is sensitive to changes. If both conditions (2) and (3) are satisfied for x_i , then the change does not propagate to variable x_i and therefore is common between the designs that satisfy functional requirements \mathbf{FR}^* and $\mathbf{FR}^* + \Delta \mathbf{FR}$. In other words, it is a platform variable for these designs.

Platform identification

The problem is to find the partitioning of the design vector $\mathbf{x} = \{\mathbf{x}_p, \mathbf{x}_c\}$ that contains the greatest number of platform variables \mathbf{x}_p (and the least number of customized variables \mathbf{x}_c), given the functional requirements \mathbf{FR}^α and \mathbf{FR}^β corresponding to differences in exogenous factors affecting the two variants α and β . Given \mathbf{x}_p , the design variables of each variant can be written as in equation (4).

$$\begin{aligned} \mathbf{x}^\alpha &= \{\mathbf{x}_p, \mathbf{x}_c^\alpha\} \\ \mathbf{x}^\beta &= \{\mathbf{x}_p, \mathbf{x}_c^\beta\} \\ \mathbf{x}^* &= \{\mathbf{x}_p, \mathbf{x}_c^*\} \end{aligned} \quad (4)$$

Consider a variant \mathbf{x}^* designed to functional requirements \mathbf{FR}^{*2} . Variant \mathbf{x}^* is essentially an unknown starting design point, from which variants are examined based on their differences in functional requirements according to the previous section. If each of the two \mathbf{x}^α and \mathbf{x}^β share the same platform variables with \mathbf{x}^* , then they also share the same platform variables.

According to condition (2), a design variable x_i may be a platform variable between \mathbf{x}^* and \mathbf{x}^α if

$$\left. \frac{\partial x_i}{\partial FR_j} \right|_* (FR_j^\alpha - FR_j^*) = 0 \quad \text{for all } j = 1 \dots m \quad (5)$$

where $\cdot|_*$ denotes that the quantity is evaluated in the neighborhood signified by $*$. If condition (2) applies as well between variants \mathbf{x}^* and \mathbf{x}^β ,

² This variant can coincide with α , β or be a completely different design conforming to either functional requirements \mathbf{FR}^α or \mathbf{FR}^β . If $\mathbf{x}^* = \mathbf{x}^\alpha$ then \mathbf{x}^α is meant to be the “base” design and \mathbf{x}^β is its evolution, and vice-versa.

$$\left. \frac{\partial x_i}{\partial FR_j} \right|_* (FR_j^\beta - FR_j^*) = 0 \quad \text{for all } j = 1 \dots m \quad (6)$$

then variant \mathbf{x}^* can be operated under either functional requirements \mathbf{FR}^α or \mathbf{FR}^β , and design variable x_i^* will not be directly impacted by this change. This is shown by subtracting condition (6) from (5):

$$\left. \frac{\partial x_i}{\partial FR_j} \right|_* (FR_j^\beta - FR_j^\alpha) = 0 \quad \text{for all } j = 1 \dots m \quad (7)$$

Similarly, condition (3), written for design variable x_i , between variants \mathbf{x}^* and \mathbf{x}^β becomes

$$\left. \frac{\partial x_i}{\partial x_s} \right|_* (x_s^\beta - x_s^*) = 0 \quad \text{for all } s = 1 \dots n \quad (8)$$

Written for design variable x_i , between variants \mathbf{x}^* and \mathbf{x}^α , condition (3) becomes

$$\left. \frac{\partial x_i}{\partial x_s} \right|_* (x_s^\alpha - x_s^*) = 0 \quad \text{for all } s = 1 \dots n \quad (9)$$

Subtracting (9) from (8) yields the condition for insensitivity of variable x_i^* to changes in any other variable in the range $\mathbf{x}^\beta - \mathbf{x}^\alpha$:

$$\left. \frac{\partial x_i}{\partial x_s} \right|_* (x_s^\beta - x_s^\alpha) = 0 \quad \text{for all } s = 1 \dots n \quad (10)$$

Together, equations (7) and (10) are the sufficient conditions for design variable x_i^* to be part of the shared platform between variants \mathbf{x}^* , \mathbf{x}^α and \mathbf{x}^β . For each j , condition (7) will be true if (a) $FR_j^\beta - FR_j^\alpha = 0$, i.e., functional requirement j does not change despite changes in exogenous factors, or (b) if the partial derivative $[\partial x_i / \partial FR_j]_*$ is zero. Therefore, platform components can only be sensitive to changes in functional requirements that are invariant to changes in exogenous factors. Likewise, condition (10) will be true for variable s if $x_s^\beta - x_s^\alpha = 0$ or $\partial x_i / \partial x_s|_* = 0$. Therefore, platform components can only be sensitive to unit changes in the design specifications of other platform components. Conditions (11) and (12) define the set of platform variables. Condition (11) says that all platform variables must be insensitive to changes in functional requirements between the variants considered. Condition (12) says that platform variables must be insensitive to customized variables for the variants considered.

$$\left. \frac{\partial x_p}{\partial FR_c} \right|_* = 0 \quad \forall c, p \quad (11)$$

$$\left. \frac{\partial x_p}{\partial x_c} \right|_* = 0 \quad \forall c, p \quad (12)$$

A sensitivity DSM of the variant \mathbf{x}^* can be partitioned to isolate the platform variables, as Figure 2 shows. The functional requirements that change between the variants are listed first, followed by the platform variables. Last are the customized design variables. Conditions (11)

and (12) imply that the blocks East and West of the diagonal block of platform variables must be equal to zero by definition.³

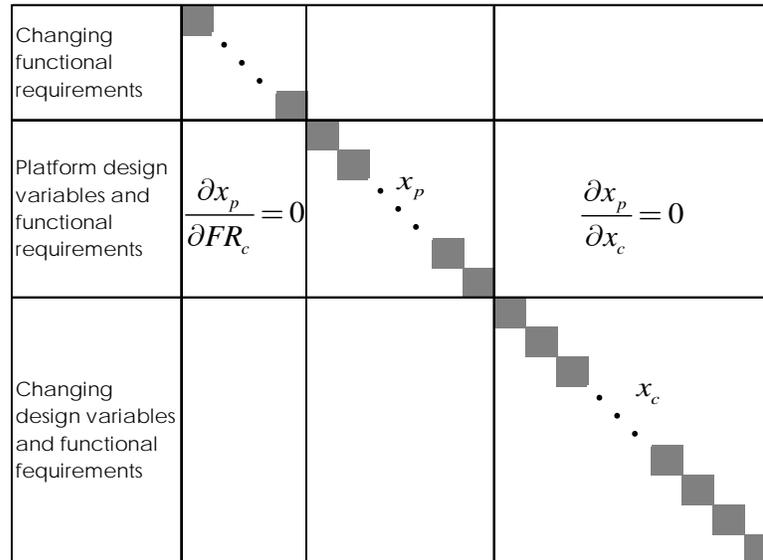


Figure 2: Invariant Design Rules on an S-DSM

The re-arrangement of the SDSM in Figure 2 shows why the platform variables provide the design rules for the variants in the product family. Design rules is the name coined by Baldwin and Clark (2000) to refer to the system components or variables that are established first in the design process and dictate the design of other components of variables. Therefore, design rules are unaffected by other variables, while at the same time they constrain the design of other variables. Platform components are thus design rules as the block on their East is zero and the block right below them, denoting sensitivity of the customized variables to platform variables, is generally non-zero.

Algorithm for platform identification

This section presents an algorithm for the identification of the largest set of platform variables. The algorithm operates on the SDSM of Figure 1 and partitions it in such a way so that the blocks East and West of the block of platform variables are equal to zero within tolerance limits (Figure 2).

Conditions (7) and (10) or equations (11) and (12) cannot be used directly for determining the platform variables between variants. If conditions (7) and (10) are satisfied, a design variable *i* is necessarily part of a platform between variants \mathbf{x}^α and \mathbf{x}^β . However, conditions (7) and (10) are only useful for *checking* whether a design variable belongs to the platform subset, not *locating* the platform subset. Also, equations (11) and (12) that hold for all platform and customized variables, are not directly useful for determining what the partitioning of the design vector should be, given the sensitivity DSM at a solution \mathbf{x}^* and the functional requirements of the variants, \mathbf{FR}^α or \mathbf{FR}^β .

Table 1 describes the algorithm steps. The algorithm involves a running list Π_k (subscript *k*

³ Or almost equal to zero, depending on the accepted tolerance.

for iteration k) of variables, initially consisting of all elements of the SDSM that are directly insensitive to the changing functional requirements. By the end of the first loop (Step 2), Π_k contains the maximum possible number of platform variables. In this Π_k , it is very likely that some variables are sensitive to changes in customized variables (not in Π_k). On a second loop, each potential platform variable is examined and removed from Π_k if it is sensitive to a customized variable. The algorithm terminates when the IDR list remains unchanged, or if Π_k is empty.

Table 1: Algorithm for the location of invariant design rules

STEP	Description	Variables
1	Establish running list of variables that are potential design rules	Π_k
2	Examine S-DSM element i . If i is not affected by changes in the changing functional requirements, then add element i to Π_k .	
3	Repeat Step 2 for next element until all elements have been examined.	
4	Store running Π_k	Π_k contains maximum set of potential design rules
5	Check each element i against each element j . If $SDSM_{i,j} = 1$ and $x_i \in \Pi_k$ and $x_j \notin \Pi_k$ then remove element i from Π_k and go to Step 6. Otherwise, examine for next element $j = j + 1$.	$\Pi_k = \Pi_k - \{x_i\}$
6	Repeat step 4 for next element i until all elements have been examined.	
7	If $\Pi_k = \Pi_{k-1}$ or $\Pi_k = \emptyset$ then the algorithm has converged; terminate. Otherwise, go to Step 4	

The algorithm in Table 1 is guaranteed to find the largest set of platform variables, not just any set. To show this, it is enough to show that Π_k at iteration k always contains the largest set of platform variables \mathbf{x}_p .

When the first loop is finished, at step 2, the running list Π_k indeed contains the largest \mathbf{x}_p . To show this, consider the complementary set to Π_k , $\bar{\Pi}_k$. $\bar{\Pi}_k$ contains all variables that do not satisfy condition (11). Since the variables in \mathbf{x}_p must satisfy both conditions (11) and (12), it follows that $\bar{\Pi}_k$ is the smallest possible \mathbf{x}_c ; therefore, Π_k contains the largest possible set of

platform variables. So, $\mathbf{x}_p \subset \Pi_k$.

The second loop (steps 4-6) starts with $\bar{\Pi}_k$, and in every iteration k an element is removed so that $\Pi_k \subseteq \Pi_{k-1}$. Because in each iteration the element removed does not satisfy condition (12) it also follows that

$$\mathbf{x}_p \subseteq \Pi_k \subseteq \Pi_{k-1} \quad (13)$$

From equation (13) it follows that the first feasible Π_k will be the largest set of platform variables.

DRAFT: Platform identification at higher system levels: application

In the final part of this paper we extend the previously presented concepts to a higher level of system aggregation, so that the process can be used as a design management tool. We illustrate such use of the concepts with an example from the automotive industry.

A variable-based DSM can be clustered in so that subsystems and components are defined (Figure 3). The clustering of a DSM is not unique: drawing boundaries around sets of design variables and considering these sets as subsystems involves a trade-off: the larger the subsystems, the fewer interactions are left outside the boundaries; the smaller the subsystems, the more interactions exist between their variables and outside their boundaries. Similarly, variables can belong to two or more systems simultaneously (so that these overlap, e.g., engine and transmission systems in Figure 3); alternatively, the common variables can be regarded as a separate “link” subsystem, interacting with both systems 1 and 2. For this paper, it is assumed that some clustering of the design variables into subsystems and components is known and accepted within the developing organization.⁴

Figure 4 shows the aggregation (clustering) of many hundreds of design variables for an automotive vehicle into 11 most important to characterize individual subsystems. Specifically, the powertrain is characterized by the fuel tank capacity (FC) and engine displacement (ED). The chassis is defined by the wheel track (WT⁵), wheel base (WB) and ground clearance (GC). Overall total length (LT) and height (HT) are associated with the body, which can be of type BOF or BFI, while the wheels are characterized by tire width (TW) and diameter (TD).

These design variables describe the subsystems that together achieve the core functionalities of an automobile: propelling, housing and towing. Propelling is the ability of the vehicle to roll on a surface as well as to accelerate and decelerate on command. The primary vehicle modules responsible for this process are the powertrain, the chassis and the wheels. The powertrain comprises, among other parts the fuel tank, engine, transmission, drive shaft and differential. The chassis is made up primarily of the structural underbody (carriage), the braking system as well as the suspension system. The wheels allow the vehicle to roll and transmit the torque generated by the engine to the road. The body of the automobile houses the passengers and cargo, thus shielding them from wind and external elements. It also reduces drag and contributes significantly to the external aesthetic appeal of the vehicle (styling). In a “body-on-frame” (BOF) architecture the chassis and body are clearly separated, whereas in a body-frame-integral (BFI) architecture they are more tightly integrated (Whitney 2004). Finally, the towing capacity (TC)

⁴ See Sharman et al (2002) and Yu et al (2003) for a discussion on clustering DSM's of subsystems and components.

⁵ We define WT as the front wheel track in this paper. The difference between front and rear wheel track is usually very small in passenger cars, but can be more significant in trucks.

of a vehicle is primarily driven by the power of the engine and the ability of the chassis to transmit the towing load from the hitch through the frame and on to the wheels. These statements reflect a mapping from internal functions to parts and assemblies. Creating this function-to-form mapping is the function of product architecture (Crawley 2001).

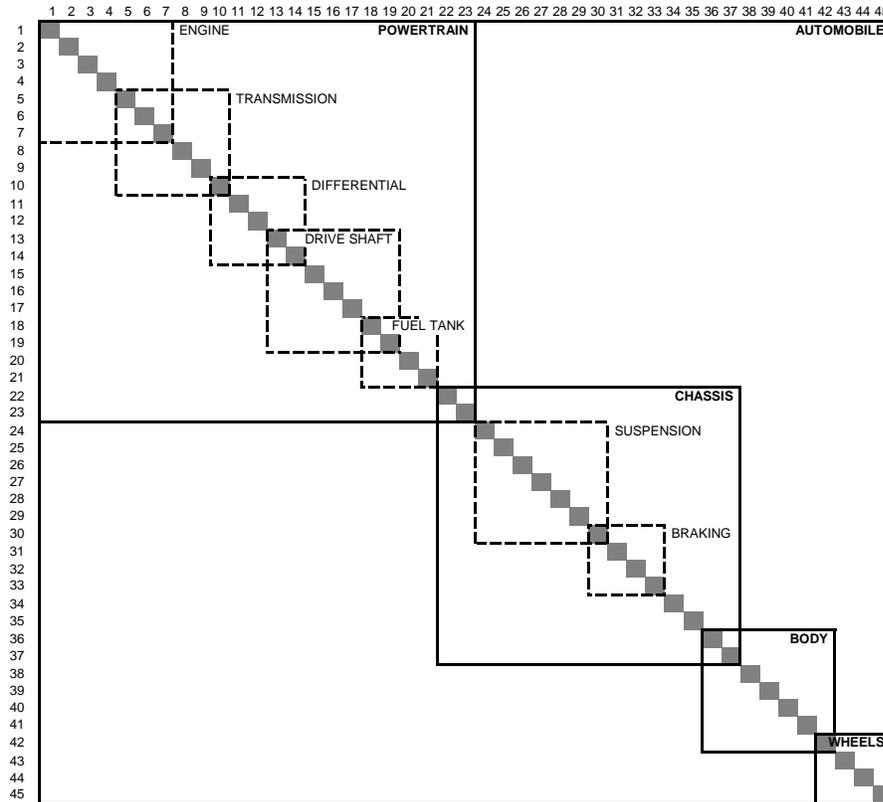


Figure 3: Clustering of a 45-variable DSM into 7 systems

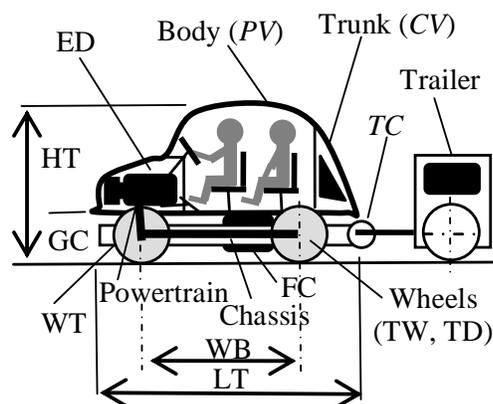


Figure 4: Vehicle subsystem-level design variables

In turn, the core functionality of an automobile can presumably be characterized by certain quantities, the functional attributes (or functional requirements), e.g., passenger volume (PV), cargo volume (CV), towing capacity (TC), fuel economy (FE) and acceleration (AC). These

functional requirements represent the value of the product system to the user or the buyer, and they will differ from one vehicle to another as vehicles are intended to satisfy different customers with different needs in different geographical and cultural settings. Unfortunately, functional requirements are seldom mapped one-to-one to design variables⁶.

In fact, functional requirements will affect (and be determined as a function of) multiple design variables as Figure 5 shows. In Figure 5 the functional requirements and aggregate design variables appear on the same DSM just as Figure 2 above. Also, interactions represent an aggregate representation of the interactions between subsystem design variables (shown in Figure 3), just as the line items in Figure 5 are an aggregate representation of the cluster of design variables in Figure 3.

FR DV	Label	PV	CV	TC	FE	AC	FC	ED	WT	WB	GC	HT	LT	TW	TD	HP	CW
FR1: Passenger Volume	PV	1							1	1		1	1				
FR2: Cargo Volume	CV		1						1	1		1	1				
FR3: Towing Capacity	TC			1					1	1	1	1				1	1
FR4: Fuel Economy	FE				1			1	1			1			1	1	
FR5: Acceleration	AC					1										1	1
X1: Fuel Capacity	FC						1										
X2: Engine Displacement	ED							1									
X3: Wheel Track	WT	1	1	1	1				1								
X4: Wheel Base	WB	1	1	1						1							
X5: Ground Clearance	GC				1						1						
X6: Height	HT	1	1	1	1							1					
X7: Length Overall	LT	1	1										1				
X8: Tire Width	TW													1			
X9: Tire Diameter	TD				1										1		
X10: Horsepower Rating	HP				1	1	1	1								1	
X11: Curb Weight	CW				1	1	1	1	1	1		1	1				1

Figure 5: Component-based DSM, extended to include functional requirements

Formulating the Sensitivity-DSM for a specific variant within the given architectural concept, again requires that the interaction in row *i* and column *j* be interpreted as “the change necessary to line item *i* because of a unit change in line item *j*. In other words, if the design variables of component *i* are not (significantly) constraining those of component *j* in the neighborhood of the specific solution in mind, then the entry *i, j* should be zero.

With this convention, all the concepts developed in the first part of the paper are transferable to component-based DSM’s; the difference is limited to the way sensitivities are quantified. In the variable-based SDSM, sensitivity is objectively defined to be the relative change in one variable as a consequence of a change in another. In the component-based SDSM, sensitivity is subjectively defined as the change necessary in one component as a consequence of change in another. In other words, since components are described by many variables, designers should simply use judgment as to whether the design of a component influences the design of another.

Multi-level platforms

A component-level SDSM in the form of Figure 5 can be partitioned according to the algorithm presented here, so that the platform subsystems are isolated from the customized ones for a collection of variants. Suppose that functional requirements 5 and 6 changed between variants, and that such a partitioning resulted in the SDSM of Figure 6, with subsystems 1, 2 and 3 as part of the platform.

⁶ A system where such mapping was possible would be an “uncoupled” design (Suh 1990).

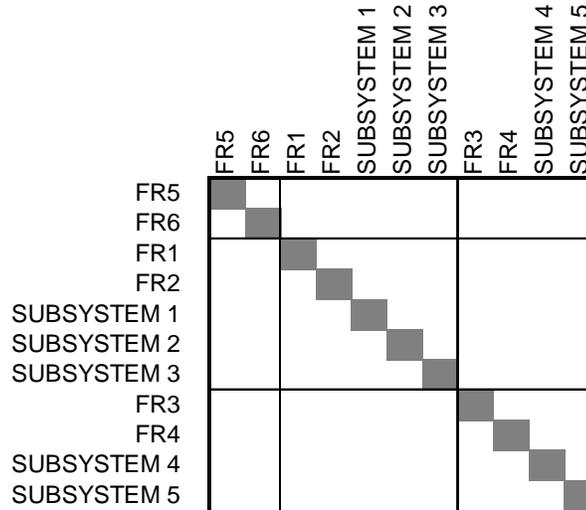


Figure 6: Component-based SDSM, partitioned to separate platform subsystems

According to this example, subsystems 4 and 5 are entirely customized between variants, whereas in reality, there may be components within subsystems 4 and 5 which can be identical. In other words, the change in the design of subsystems 4 and 5, necessary to accommodate the difference in functional requirements may be accomplished by only changing parts of these subsystems, not the entire subsystems.

Consider for example subsystem 4, and assume it is composed of 4 components. By exploding subsystem 4 into its components, and accordingly filling out the sensitivity information in the SDSM, it is possible to re-run the algorithm above to identify platform components within subsystem 4 (Figure 7 and Figure 8).

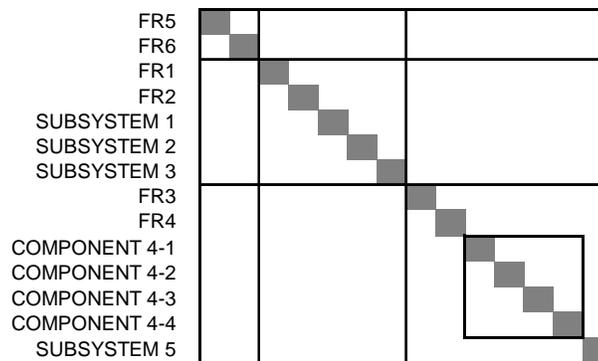


Figure 7: Partitioned SDSM, with subsystem 4 exploded into components

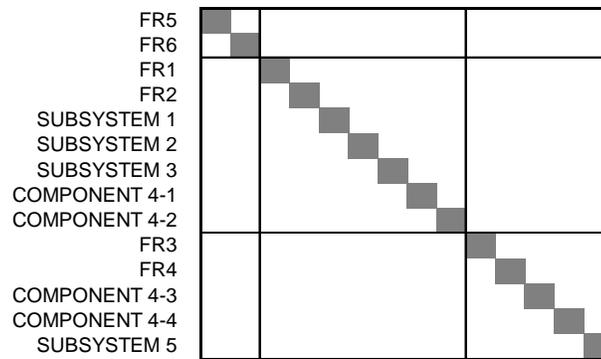


Figure 8: Re-partitioned SDSM, with platform including subsystems and components

The component-based implementation of this methodology enables a multi-disciplinary team to examine...

[To be completed]

Discussion

[To be completed]

References

- Baldwin, C.Y. and Clark, K.B. Design Rules Vol. 1 The Power of Modularity. MIT Press, Cambridge MA, 2000.
- Biegler, L., Grossmann, I. and Westerberg, A. Systematic methods of chemical process design, Prentice Hall international Series in the Physical and Chemical Engineering Sciences, Upper Saddle River, N.J,1997. ISBN: 0-13-492422-3
- Browning, T.R., Applying The Design Structure Matrix To System Decomposition And Integration Problems: A Review And New Directions. IEEE Transactions On Engineering Management 48(3), August 2001 pp. 292-306
- de Weck, O.L., Determining Product Platform Extent, pp. 241-301 in "Product Platform and Product Family Design, Methods and Applications," Simpson T.W., Siddique, Z. and Jiao, J. (Editors). Springer Science, New York, NY 2006. ISBN: 0-387-25721-7
- Crawley E., "Systems Architecture", Course Notes ESD.34/16.882, Massachusetts Institute of Technology, 2001
- Eckert, C., Clarkson, P.J. and Zanker, W. Change and customization in complex engineering domains. Research in Engineering Design 15, 2004 pp. 1-21
- Eppinger, S.D. and Salminen, V. Patterns of product development interactions. International Conference on Engineering Design, Glasgow, August 21-23, 2001
- Fricke, E. and Schultz, A.P.. Design for Changeability (DfC): Principles to Enable Changes in Systems Throughout their Entire Lifecycle. Systems Engineering 8(4), 2005, pp. 342-359
- Gonzalez-Zugasti, J.P., Otto, K.N. and Baker, J.D. Assessing value in platformed product family design. Research in Engineering Design 13, 2001, pp. 30-41.
- Kokkolaras, M., Fellini, R., Kim, H.M. and Papalambros, P.Y. Analytical Cascading in Product Family Design, pp. 226-240 in "Product Platform and Product Family Design, Methods and Applications," Simpson T.W., Siddique, Z. and Jiao, J. (Editors). Springer Science, New

- York, NY 2006. ISBN: 0-387-25721-7
- Martin, M.V. and Ishii, K. Design for Variety: Developing Standardized and Modularized Product Platform Architectures. *Research in Engineering Design* 13, 2002. pp. 213-235,
- Meyer, M.H. and Lehnerd, A.P. *The Power of Product Platforms*, The Free Press, New York 1997
- Parkash, S., *Refining processes handbook* ISBN: 0-7506-7721-X, Elsevier Publishing, Boston MA, 2003
- Pimmler, T. U. and Eppinger, S. D., *Integration Analysis of Product Decompositions*, in Proc. ASME 6th Int. Conf. on Design Theory and Methodology, Minneapolis, MN, 1994.
- Sharman, D.M., Yassine, A.A. and Carlile, P. Characterizing modular architectures. *Proceedings of International Design Engineering Technical Conferences ASME 2002, Design Theory Methodology Conference*, Montreal, Canada, September 29-October 2, 2002
- Simpson T., *Product platform design and customization: status and promise*. *Artificial intelligence for Engineering design, analysis and manufacturing* 18, 2004, pp.3-20
- Simpson, T.W. and D'Souza, B. Assessing variable levels of platform commonality within a product family using a multiobjective genetic algorithm. *9th AIA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 4-6 September 2002, Atlanta, Georgia.
- Simpson, T.W. *Methods for Optimizing Product Platforms and Product Families*, in "Product Platform and Product Family Design, Methods and Applications," Simpson T.W., Siddique, Z. and Jiao, J. (Editors). Springer Science, New York, NY, 2006, ISBN: 0-387-25721-7
- Steward, D. *Planning and Managing the Design of Systems*, *Proceedings of the Portland International Conference on Management of Engineering and Technology*, Portland, OR, USA 27-31 October 1991.
- Steward, D. *System Analysis and Management: Structure, Strategy and Design*, Petrocelli Books, New York, 1981
- Suh, E.S., *Flexible Product Platforms*, PhD Thesis, Massachusetts Institute of Technology, Engineering Systems Division, Cambridge MA, 2005
- Suh, N., *The principles of Design*. Oxford University Press, New York 1990
- Sullivan, K.J., Griswold, W.G. and Ben Hallen, Y.C. *The Structure and Value of Modularity in Software design*. *Proceedings, ESEC/FSE Conference*, Vienna, Austria, 2001.
- Ulrich, K., *The role of product architecture in the manufacturing firm*, *Research Policy* 24, 1995
- Yassine, A.A. and Falkenburg D.R., *A Framework for Design Process specifications management*, *Journal of Engineering Design* 10(3), 1999 pp. 223-234
- Yu, T.L., Yassine, A.A and Goldberg D.G. *A Genetic Algorithm For Developing Modular Product Architectures*, *Proceedings of DETC '03, ASME 2003 International Design Engineering Technical Conferences. Computers and Information in Engineering Conference* Chicago, Illinois USA, September 2-6, 2003
- Whitney, D., *Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development*, Oxford University Press, 2004