Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713641621

Optimal component sharing in a product family by simultaneous consideration of minimum description length and impact metric

Alvaro J. Rojas Arciniegas[a]; Harrison M. Kim[a]
[a] Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA

First published on: 11 October 2010

To cite this Article Rojas Arciniegas, Alvaro J. and Kim, Harrison M.(2011) 'Optimal component sharing in a product family by simultaneous consideration of minimum description length and impact metric', Engineering Optimization, 43: 2, 175 — 192, First published on: 11 October 2010 (iFirst)
To link to this Article: DOI: 10.1080/0305215X.2010.486032
URL: http://dx.doi.org/10.1080/0305215X.2010.486032

Engineering Optimization
Vol. 43, No. 2, February 2011, 175–192

Taylor & Francis
Taylor & Francis Group

Optimal component sharing in a product family by simultaneous consideration of minimum description length and impact metric

Alvaro J. Rojas Arciniegas and Harrison M. Kim*

Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, 104 S. Mathews Avenue, Urbana, IL 61801, USA

(*Received 5 November 2009; final version received 21 March 2010*)

The design of an architecture for a family of products is a challenging task. Traditionally, only a single product case is considered when optimization techniques are used to define the optimum clustering of the components, or in other words the module definition for the product. This article proposes a framework to analyse multiple products of a given family and achieve an optimal module definition considering component sharing across some or all the products. The optimal sharing decision process is implemented with a dual objective function: a 'minimum description length' representation of the overall product architecture, and the 'impact metric' description of the selected components to share. An illustration example is developed for three digital cameras of the same brand and the results reveal the recommended set of components to share across the family as well as the optimal architecture for each one of the products.

Keywords: product family design; module definition; component sharing; product architecture

1. Introduction

In the field of product design, product architecture has been defined as 'the scheme by which the function of a product is allocated to physical components' (Ulrich 1995). Similarly, when a set of products (also known as variants) is developed from a common set of subsystems, modules and/or components (which constitute the product platform), the products are considered a product family (Meyer and Lehnerd 1997, Khajavirad *et al.* 2009). When designing a family of products, defining the architecture is a key decision to make since it affects not only the products being developed at that time, but also the variants to develop down the road from the given platform. Therefore, the decisions of which components should form a module or which should be shared across some or all the products in the family require careful consideration.

In the case of single products, many of the studies of product architecture have used the Design Structure Matrix (DSM) to model the product, as pointed out by Browning (2001). The DSM was introduced by Steward (1981) and it has been widely used as a way to represent the interconnections of a product, whether it means geometrical joints, electrical connections, material

*Corresponding author. Email: hmkim@illinois.edu

ISSN 0305-215X print/ISSN 1029-0273 online

DOI: 10.1080/0305215X.2010.486032
http://www.informaworld.com

flow, etc. This representation of the product has allowed researchers and designers to find the optimal module definition by clustering the cells with strong relationships among components through manipulation of the rows and columns of the matrix (*e.g.* Yu *et al.* 2007). However, when multiple products are considered, trying to capture the relationships, similarities or differences among the various products through the use of multiple DSMs is not an easy task. Every product has its own set of components; therefore each DSM is different in size and content, making it very difficult to relate to other DSMs.

Other approaches rely on a commonality metric which indicates the number of unique components of a product relative to the total number of components (*e.g.* Martin and Ishii 1996, Mikkola and Gassmann 2003). Even when this metric involves only counting the number of components, a very important aspect is captured in a product family: 'commonality'. Simpson (2004) highlights that the main objective when creating a platform is to facilitate the generation of new products by having a common structure from which the new variants can be developed. Therefore, the question is not *how many* components should be common or standard, but rather *which* components should be shared across the family. The difference is subtle, but of great significance: it is more important to find *the set of* components to have in common, rather than finding *a number* of components to share.

Different criteria have been used in order to determine *the* components to share: cost considerations (*e.g.* Browning and Eppinger 2002, Zacharias and Yassine 2007, Moon *et al.* 2008), Bill of Materials (BOM) (*e.g.* Steva *et al.* 2006), product attributes (*e.g.* Tucker and Kim 2008), environmental concerns (*e.g.* Dahmus and Gutowski 2007, Kwak *et al.* 2007, Pandey and Thurston 2008), product design variables (*e.g.* Khajavirad and Michalek 2008, Khajavirad *et al.* 2009), etc. However, little attention has been paid to the way in which the components are arranged into the products. The attempt of this article is to provide guidance to the designer/engineer on which components should be shared across the products of a given family based on the product architecture.

The framework proposed in this article goes further than previous approaches which only consider optimal clustering for a single product DSM (*e.g.* Wang and Antonsson 2004, Wang 2007, Yu *et al.* 2007) and considers multiple DSMs simultaneously in order to find the set of components to share across the product family. The decision is based on the connectivity between components, along with the functional decomposition from each one of the products.

The remaining sections of the article proceed as follows: a review of the relevant literature in this topic is presented in Section 2; the sharing decision framework is introduced in Section 3; which is followed by its application to a case study in Section 4; the results of the case study and its relevance are discussed in Section 5; and the article concludes with Section 6, which summarizes the proposed framework and the implications of the observed results.

2. Literature review

Finding a way to represent a product, capture its relationships and to be able to analyse similarities, dependencies, component clustering and modularity, have been some of the areas related to product architecture that have been greatly explored. In terms of product representations, three tendencies tend to dominate the approaches: a graph/network representation (*e.g.* Wang and Antonsson 2005, Sosa *et al.* 2007), matrices (*e.g.* Steward 1981, Browning 2001) and commonality indices (*e.g.* Thevenot and Simpson 2006). The first two are intrinsically related since it has been shown that graphs can be represented by matrices and vice versa (Browning 2001, Wang 2007). The Design Structure Matrix (DSM) is a widely accepted approach for a matrix representation and will be reviewed briefly in Section 2.1.

The modularity of an architecture is understood as the degree to which the product functions are implemented by the physical elements of the product (Gershenson *et al.* 2003, Ulrich and Eppinger 2004). Multiple metrics have been developed to quantify this concept in a particular manner, most of them reflecting that a 'good' module has strong internal connections and weak external connections (Gershenson *et al.* 2004). This idea is reflected by the Minimal Description principle, which has been used to find the optimal clustering for the product, and the Impact Metric, which provides a description of the easiness to change a component in a given architecture. These two methods will be reviewed in Sections 2.2 and 2.3, respectively, since these are key aspects to consider when trying to decide which components should be shared across a family of products.

2.1. *Design structure matrix*

The Design Structure Matrix (DSM) introduced by Steward (1981) has been well established as a tool for system analysis since it captures the relationship between elements. 'A DSM provides a simple, compact and visual representation of a complex system' – Browning (2001) – and it has been applied to model products, processes, tasks and organizations (*e.g.* Eppinger *et al.* 1994, Browning 2001, Browning and Eppinger 2002). In the case of product design the DSM arranges the components of the product in both rows and columns of a matrix, and each cell contains the relationship between the corresponding components.

The most basic DSM is binary (see Figure 1) in which the *X*'s or *ones* represent a relationship or connection between the corresponding components. The DSM is also able to capture directionality in the relationships. Usually the components in the columns are the suppliers and those in the rows are the receivers, and in that case, the DSM is not necessarily symmetric. Given the flexibility of this representation, there are many different adaptations depending on the information to be collected from the product. Pimmler and Eppinger (1994) and Smaling and de Weck (2007) use the DSM to specify whether the nature of the relationship between components is a physical connection, energy flow, mass flow or information flow. Similarly, Martin (1999) and Martin and Ishii (2002) introduced the Coupling Indices, which used a DSM that contains the sensitivity analysis for specification changes in a product.

Overall, the DSM is a very effective tool to represent the relationships inside a product. However, when multiple products are considered, there is no representation established yet, and it is difficult to extract similarities or differences between products from multiple DSMs.

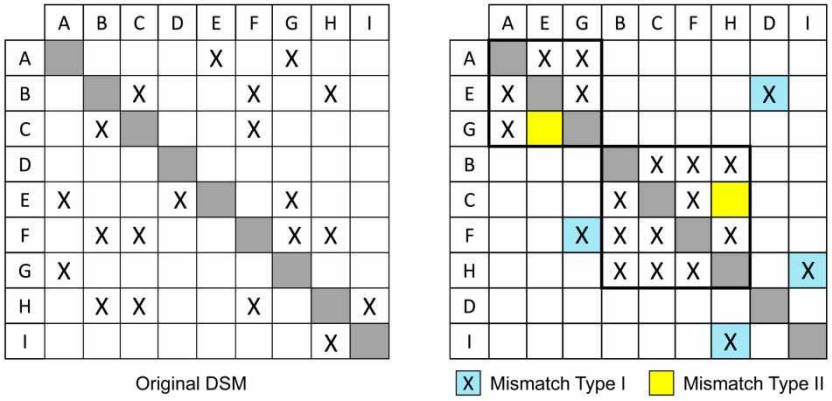

Figure 1. Binary DSM for a fictitious product unordered and rearranged for clustering.

2.2. *Minimal description length*

A different approach to modelling a product structure is through the use of the minimal description length (MDL). According to Yu *et al.* (2007) the goal of the minimal description principle is to represent a system uniquely, in the simplest possible way.

In the case of product architecture, a binary DSM can be represented by the number of clusters it has and the elements that belong to each cluster. In Figure 1, the product of nine components can be described by stating that it has two clusters, one with components {A,E,G} and other with components {B,C,F,H}. However, that description is not accurate since the cells outside the clusters are not all empty (mismatch Type I) and the cells inside the clusters are not fully populated (mismatch Type II); therefore, the representation is only complete when these mismatches have been taken into account.

Nevertheless, there would be many possible ways to represent a given DSM depending on the number of clusters or the selected components for each cluster. Each representation would be a valid one, but the length of the descriptions would be different. A more complicated model would require more details and thus a longer description. Therefore, the MDL principle can be interpreted as trying to find the shortest of those valid descriptions for the data (Wang and Antonsson 2004, 2005).

Yu *et al.* (2007) used this approach to represent a DSM and find the optimal clustering strategy. In their implementation, the length of the model is captured according to Formula (1) and the length of the mismatches is captured by Formula (2):

$$n_{\mathrm{cl}} \log_2 n_{\mathrm{c}} + \log_2 n_{\mathrm{c}} \sum_{i=1}^{n_{\mathrm{cl}}} cl_i, \tag{1}$$

$$S_1(2 \log_2 n_{\mathrm{c}} + 1) + S_2(2 \log_2 n_{\mathrm{c}} + 1). \tag{2}$$

In these formulae, n_{cl} represents the number of clusters, n_{c} is the total number of components of the product, and cl_i is the number of elements in cluster i. In Formula (2) the term S_1 corresponds to the number of non-empty cells outside the clusters defined by the model (mismatch Type I) and the term S_2 corresponds to the number of empty cells inside the clusters (mismatch Type II) excluding the elements of the main diagonal which are not considered as mismatches. This definition works only when the DSM is binary; when the elements in the DSM are not binary entries, the mismatches (according to Yu *et al.* 2007) should be redefined as follows:

$$S_1 = \sum_{d'_{ij}=1} (1 - p_{ij}); \quad S_2 = \sum_{d'_{ij}=0} (p_{ij}); \tag{3}$$

$$p_{ij} = \frac{d_{ij} - d_{\min}}{d_{\max} - d_{\min}}; \quad d_{\max} = \max_{ij} d_{ij}; \quad d_{\min} = \min_{ij} d_{ij}; \tag{4}$$

where d_{ij} is the entry of the ith row and jth column of the DSM, p_{ij} is the same entry normalized and d'_{ij} is the entry for the ideal binary DSM described by the model without considering any of the mismatches.

In the end, the length of a representation for a given DSM can be obtained applying the formula

$$f_{\mathrm{MDL}} = (1 - \alpha - \beta) \left(n_{\mathrm{cl}} \log_2 n_{\mathrm{c}} + \log_2 n_{\mathrm{c}} \sum_{i=1}^{n_{\mathrm{cl}}} cl_i \right) + \alpha[S_1(2 \log_2 n_{\mathrm{c}} + 1)] + \beta[S_2(2 \log_2 n_{\mathrm{c}} + 1)], \tag{5}$$

where α and β are weights between 0 and 1, making the function a linear combination of three terms, the length of the model and the length of the mismatches type I and II.

In the work by Yu *et al.* (2007), this becomes the objective function (min f_{MDL}) for the optimization in which the optimal clustering is found. The representation of the entire DSM observed in Formula (5) is intended for product-level analysis; hence, no consideration has been given to multiple products, component description or component sharing, which are key aspects of product family design.

2.3. *Impact metric*

Rojas and Esterman (2008) and Rojas-Arciniegas (2008) proposed a metric to capture the impact to change a component in a given platform, combining the MDL representation for each component and the Coupling Index (CI) score. The MDL formulation used in the IM is different given that the objective is to find a representation for the components rather than the entire DSM. The MDL formulation was derived from Wang and Antonsson (2004, 2005) and Wang (2007), in which each component is represented by the interconnections it has with other components/clusters and it is compared to the total number of connections in the system at the same level. 'Level' in this context is understood in the following manner: inside a module, a component is at the same level with other components or submodules that are in it. When there exists a connection from a component inside a module to a component outside the module, the component is considered an interface and it is compared against other interfaces at the same level.

This means that for a unit j (either component, module or interface) the description length would be given by

$$\mathrm{MDL}_j^{(\mathrm{u})} = -\log_2\left(\frac{N_j^{(\mathrm{u})}}{\sum_{k=1}^{N^{(\mathrm{u})}} N_k}\right), \tag{6}$$

in which $N_j^{(\mathrm{u})}$ is the number of units connected to unit j and $N^{(\mathrm{u})}$ is the number of units at the level in which the unit j is. Since a unit can be seen as a component if it is considered inside a module, but also as an interface if it has connections outside the module, the total representation of unit j in product k would be

$$\mathrm{MDL}_j^k = \mathrm{MDL}_j^{(\mathrm{c})} + \mathrm{MDL}_j^{(\mathrm{o})} = -\sum_{u=\{\mathrm{c},\mathrm{o}\}} \log_2\left(\frac{N_j^{(\mathrm{u})}}{\sum_{k=1}^{N^{(\mathrm{u})}} N_k}\right), \tag{7}$$

where 'c' represents components and 'o' interfaces.

In order to obtain the IM for each component, it is necessary to obtain the Coupling Index (CI). Therefore, instead of using a binary DSM (which would be enough to calculate the MDL for the components), the DSM contains the sensitivity analysis for the impact of a change in the specifications of the components (using a scale {9-6-3-1} in which 9 represents that the receiving component is highly sensitive to a change in specification while 1 represents low sensitivity – Martin 1999). This assumes that the components in the columns supply the specification and the components in the rows receive the specifications.

Each row and column of the DSM is added giving the CI-Receiving (CI-R) for the rows and the CI-Supplying (CI-S) for the columns. The total CI for each component is the sum of the CI-S

and the CI-R (see Equation 8 for the CI of component j in product k):

$$\mathrm{CI}_j^k = \mathrm{CI\text{-}S}_j^k + \mathrm{CI\text{-}R}_j^k = \sum_{i=1}^{n_c^k} \mathrm{DSM}^k(i, j) + \sum_{i=1}^{n_c^k} \mathrm{DSM}^k(j, i). \tag{8}$$

The last step in the calculation of the IM for each component is to multiply the MDL representation by the CI (see Equation 9). This scaling effect serves well to leverage the architectural description of each component based on the number of interconnections as well as the strength of those connections or coupling of the components. It also highlights those components which would be more difficult to change (a higher IM rate indicates that the component is much more connected with the rest of the system and/or the connections are strong; hence, other components would get affected by a change in that particular component):

$$\mathrm{IM}_j^k = \mathrm{MDL}_j^k \times \mathrm{CI}_j^k. \tag{9}$$

This score is a good way to distinguish between components which are easy to change and those which are difficult. However, there is no standard way to compare the scores for components of different products since the application of the IM to analyse multiple products has not been considered before.

3. The sharing decision and optimal clustering framework

This section presents an approach to alleviate the problem of determining the optimal set of components to share across the products in a family while achieving an optimal clustering of their components. The goal is to identify the components to be shared across multiple products of the family in an automated manner, based on the architectural information of the products and the individual components.

3.1. *Framework overview*

The proposed framework is an iterative process and it identifies candidates for component sharing based on the functional description of each component and its IM score, which represents the ease of changing the component under the *current* architecture.

The decision in component sharing is marked in binary vectors (one for each product) containing as many elements as the number of components for the product, 1 (one) for a component to be shared and 0 (zero) for a component that will be kept unique for the product. If, for example, a product with six components shared the second and fourth components with other products, the corresponding decision vector would be [0 1 0 1 0 0].

Once the decision vectors have been established, an optimization problem is solved for each individual product of the family with a dual objective: on one hand, the product MDL representation of Formula (5) finds a compact arrangement of the components in clusters; meanwhile, the decision vector is multiplied by the IM vector from Equation (9), thus selecting the components to include in the objective function and allowing those components to be easily changed in the product. At each iteration a new component arrangement is achieved for each product, progressively establishing the final list of components to share. The process is repeated until the change in the decision vectors for all the products being considered is below a given tolerance ϵ

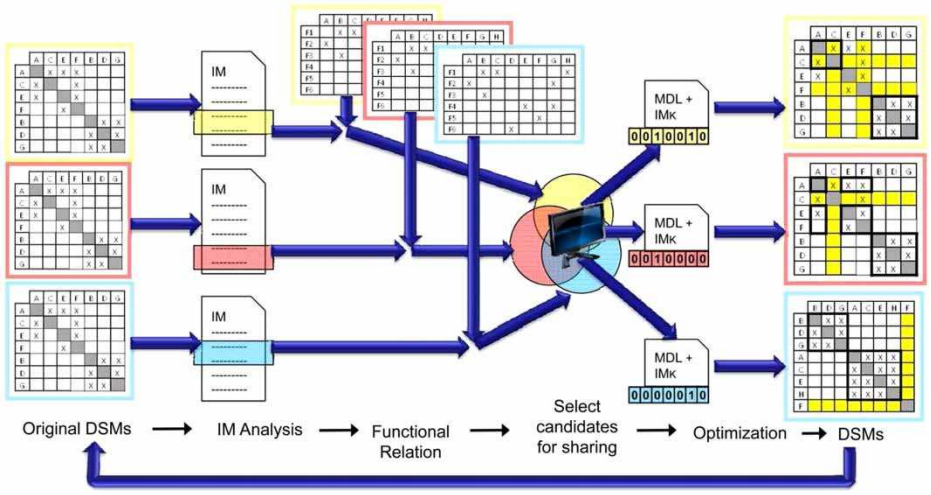

Figure 2. Proposed framework for component sharing selection and optimum clustering for multiple products.

(see Equation 10). An overall view of the framework can be seen in Figure 2.

$$\sum_{i=1}^{m} \frac{\|Y^i(k) - Y^i(k-1)\|^2}{n_c^i} \leq \epsilon \tag{10}$$

The major contribution of this framework is the automation of the optimal sharing decision process. However, one of the major concerns in achieving this goal is to identify the components of different products that can be considered as candidates for sharing. When considering multiple products, the name of the component and its material or dimensions are not always helpful for identifying similar components; hence, the proposed attempt is to look for components that fulfil identical functions.

3.2. *Functional matching*

Since the products in the family may share many functions they fulfil, it is necessary to construct a functional description for each product, in which the components can be related to the functions they fulfil. Each product has its own set of functions and components. For the purpose of the analysis it is necessary to construct an enumerative list of functions across all the products, which serves as a reference list to determine which function is common or unique.

A Function–Component Matrix (FCM) relates functions and components in a similar manner to a binary DSM (Strawbridge *et al.* 2002); each marked cell indicates that the corresponding function is fulfilled partially or completely by the corresponding component. Each component (columns in the FCM) must have a relation with at least one function (rows in the FCM), since every component has a reason to be a part of a product. Similarly, each function should have at least one relation with a component in the entire family, since there may be features that are unique to a particular product in the family.

In Figure 3 three products with a list of five functions are examined. First, looking at products A and B only, it is clear that component 1 in product A, as well as component 2 in product B, are fulfilling function 2. Therefore, they are a match (one-to-one components for a single function). Similarly, components 3 and 4 of product A match component 1 of product B that corresponds to function 3 (one-to-many components for a single function). Meanwhile, component 2 of product

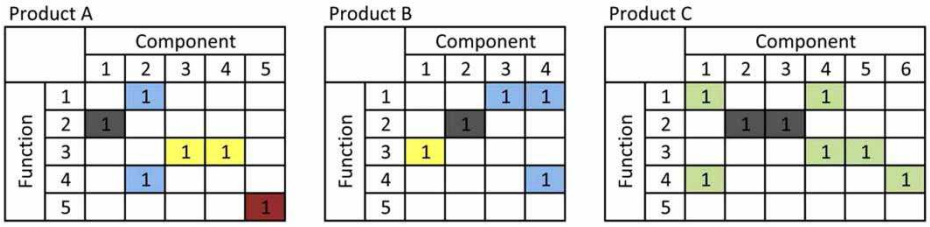

Figure 3. Example of function–component matrices for products A, B and C with five functions and functional matching.

A must match with components 3 and 4 of product B since they are fulfilling both functions 1 and 4 (one-to-many components for multiple functions). Component 5 of product A could not be matched with any component from product B since function 5 is unique for product A.

When matching components for products A and C based on the functional description, component 1 of product A is a match for components 2 and 3 of product C since they fulfil function 2. Component 5 of product A cannot be matched with any component of product C since function 5 is unique for product A. Nevertheless, when the remaining components are examined there is no way to match them individually since the functions they fulfil make them interconnected. Hence, the result is that components 2, 3 and 4 of product A match components 1, 4, 5 and 6 of product C since they fulfil functions 1, 3 and 4 (many-to-many components for multiple functions).

3.3. *Automated selection of shared components*

Once all the cases are identified, the strategy for matching the components based on the functional description is developed for pairwise comparisons in which:

- a master function-vector (f-v) keeps track of the functions evaluated for the pair;
- a local function-vector marks the functions fulfilled by the matching components;
- two component-vectors (c-vs) mark the components in each product that are necessary to fulfil the functions in the local function-vector; and
- two decision-vectors are returned after the evaluation and indicate the components to share for each product.

The pairwise selection of candidates for sharing is conducted as follows:

(1) start by evaluating the first function;
(2) mark the master f-v with the corresponding function;
(3) initialize the local f-v and c-vs,
(4) follow the row of the Function–Component Matrix (FCM) corresponding to the function being evaluated until a 1 is found;
(5) mark the corresponding component in the c-v;
(6) follow the column of the corresponding component and mark the functions that it fulfils in the local f-v;
(7) return to step 4 until all the components in both products have been evaluated;
(8) mark the functions marked in the local f-v into the master f-v;
(9) evaluate if the summation of the IM of the components marked in the c-v for each product is below a threshold previously defined; if so, mark the components in the c-vs into the decision-vectors; and
(10) continue the evaluation of the next function unmarked in the master f-v and return to step 2.

Once a pairwise evaluation has been conducted the process is repeated with all the possible pairs of products. The result of this stage is the decision vectors for all the products in the family containing the components to share from each product.

3.4. *Iterative approach*

The decision vectors are passed to the individual optimizations of the product architecture in which the optimum clustering is found for all the products (considering the set of components to share among the family, which were selected in each decision vector). However, the new arrangement of components alters the IM score from which the candidates for sharing were selected. Therefore, the new architecture becomes the initial step for the process and the evaluation is conducted again until a stopping criterion is met (see Equation 10).

One important factor in the automated selection process is the threshold value for the IM set by a decision maker; a higher value implies that more components will become candidates for sharing if the functional matching is achieved. However, this may be an important strategy for the company either to allow greater commonality among the family or decide to share only few components that will not implicate much effort to change according to the architectural information.

The overall framework proceeds as follows:

(1) construct the product DSM for each variant of the family;
(2) construct the functional relationship matrix for each product and include all the functions in the family;
(3) calculate the IM for all the components in all the products;
(4) run the component sharing selection algorithm to obtain the decision vectors for each product in the analysis;
(5) run the optimization for each product separately including both the MDL representation of the product and the IM of the selected components for sharing to obtain a new arrangement of the components in the products considering component sharing; and
(6) verify if a stopping criteria is met (*i.e.* no further change in all the decision vectors); if not, return to step 3.

The process is summarized in Figure 4.

3.5. *A genetic algorithm implementation*

The optimization is implemented by the use of genetic algorithms similar to the one proposed by Yu *et al.* (2007), in which the chromosome represents the clusters and its elements out of the components of the product. The chromosome is a binary string of $n_{\mathrm{c}}^{i} \times mn_{\mathrm{cl}}^{i}$ elements; for example, if product i has nine components ($n_{\mathrm{c}}^{i} = 9$) and a maximum of two clusters ($mn_{\mathrm{cl}}^{i} = 2$) then the chromosome would have 18 elements. An example of a chromosome is shown in Figure 5.

Since the number of components is specified in the DSM, the only additional parameter to specify is the maximum number of clusters. That implies that there may be some empty clusters and the actual number of clusters (non-empty) is automatically determined by the GA.

The objective function is a dual objective: first, the MDL representation of the product is obtained following Formula (5), which provides a global score of how compact the structure is; secondly, the decision vector is multiplied by the IM vector adding the IM score of the selected components for sharing into the objective function. The formulation is as follows:

$$f(\mathrm{DSM}^{i}, X^{i}) = f_{\mathrm{MDL}}^{i}(\mathrm{DSM}^{i}, X^{i}) + [Y^{i}(k)^{\mathrm{T}} \times IM^{i}(\mathrm{DSM}^{i}, X^{i})], \tag{11}$$

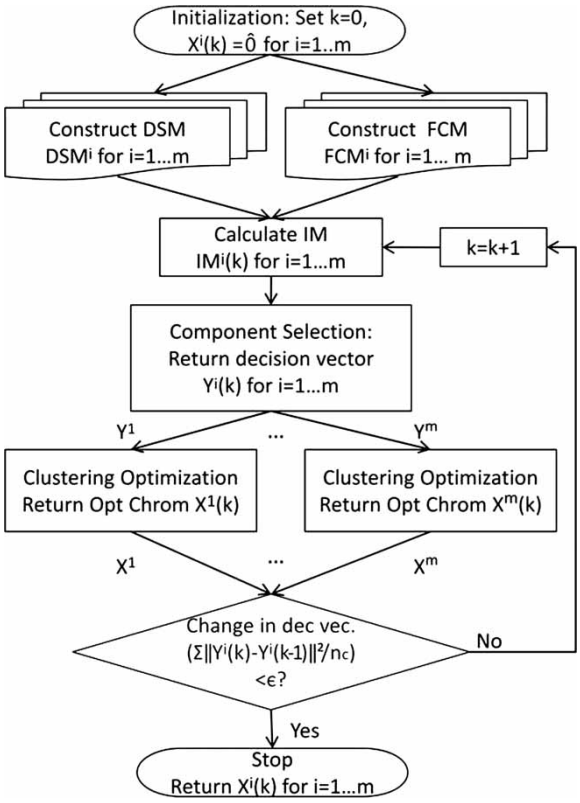

Figure 4. Framework process diagram.

Component:	A	B	C	D	E	F	G	H	I
Cluster 1:	1	0	0	0	1	0	1	0	0
Cluster 2:	0	1	1	0	0	1	0	1	0

	Cluster 1									Cluster 2								
Chromosome:	1	0	0	0	1	0	1	0	0	0	1	1	0	0	1	0	1	0

Figure 5. Example of a chromosome for the DSM in Figure 1.

in which X^i is the chromosome that is being evaluated. This formulation preserves the tight clustering achieved by the MDL and allows the consideration of sharing components by including their IM score.

4. Illustrative example

In order to illustrate the application of the proposed framework, an example was developed for three digital cameras from the Sony DSC (Digital Still Capture) portfolio.

4.1. *Digital cameras*

In the market of digital cameras most of the brands offer an entry level family of compact point-and-shoot cameras, a set of ultra-slim point-and-shoot cameras, higher end high-zoom cameras and the professional level Digital Single-Lens Reflex (DSLR) cameras. The first two classes offer many similarities both in size and features as well as the customer groups and their uses. Therefore, conducting an analysis for component sharing among the point-and-shoot cameras was a logical choice.

Sony in particular has had two lines of cameras in the first category (compact point-and-shoot), the S family and the W family. The cameras from the S family were the entry level to the cyber-shot digital cameras (the line was recently discontinued) and featured the basic functions of a point-and-shoot digital camera: easy to use, auto focus, several exposure modes, extendable optical zoom, flash, video capture with sound, among others. The cameras from the W family had been very similar but usually included slightly better features than the cameras from the S family, such as optical stabilization or a viewfinder in order to attract a more demanding customer with very similar exterior design and size. The set of ultra-slim point-and-shoot cameras has been covered by the T family, which featured a lens cover that slides, a non-extendable zoom and a very sleek design, along with most of the features from the other families. This camera is intended for a different customer and has a slightly higher price compared to a camera from the other two families.

Given the similarities and unique features of these types of camera, an illustrative study was developed with three cameras: DSC-W100, DSC-S730 and T30 (one from each line) from the available service manuals at Sony (2006b, 2007, 2006a). Figure 6 shows an image of the cameras selected for this case study.

The disassembly instructions along with the information in the manuals were analysed in order to construct a DSM for each product as well as the Function–Component Matrix (FCM) (the initial DSMs as well as the FCMs for the digital cameras are not shown due to space limitations). The cells in the DSMs were populated according to the CI specification sensitivity analysis in order to calculate the CI and IM afterwards.

4.2. *Framework application*

Since the DSMs and FCM were already developed (steps 1 and 2 of the framework), the framework continues by calculating the IM for all the components in all the products included in the analysis. The IM calculation implies a component arrangement, hence an initial chromosome was generated in which no cluster was defined (a 1-by-n_c^i zero vector was used). IM(k) would correspond to the IM vector at the kth iteration for each camera (see Figure 7).

The next step is to run the component sharing selection algorithm which performs pairwise comparisons among all the products to determine component matching (according to the functional description of the FCM) and evaluates if the sum of the IM score of the matching elements is

Figure 6. Sony digital cameras used for the case study: (a) DSC-W100 (Sony 2006b); (b) DSC-S730 (Sony 2007); and (c) DSC-T30 (Sony 2006a).

DSC-W100 Component		IM(3)	Y(3)
Ornamental Ring	1	0	1
Cabinet Front	2	511	0
LCD Window	3	140	0
Buttons	4	0	1
Side Lid	5	123	0
Cabinet Rear	6	532	0
Side Cover	7	70	0
Control Switch	8	0	1
LCD	9	233	0
LCD Plate	10	95	0
LCD PCB	11	155	0
Side Frame 1	12	153	0
Shoot Button	13	0	1
Zoom Slider	14	0	1
Mic	15	105	0
Speaker	16	0	1
Top Frame	17	255	0
Side Frame 2	18	252	0
Capacitor Holder	19	12	0
Flash	20	0	1
Capacitor	21	48	0
Flash PCB	22	319	0
CCD	23	0	1
Filters	24	175	0
Barrier Ring	25	140	0
Ornamental Ring	26	0	1
Visor	27	288	0
Lens Sub-ASM	28	682	0
Rear PCB	29	283	0
Top Flex PCB	30	222	0
Lid	31	123	0
Front PCB	32	343	0
Battery Holder	33	355	0
Main Body	34	741	0

DSC-S730 Component		IM(3)	Y(3)
Cabinet Ornamental	1	0	1
Cabinet Front	2	660	0
LCD Window	3	79	0
Buttons	4	0	1
Control Switch	5	0	1
Cabinet Rear	6	633	0
LCD	7	486	0
LCD Plate	8	156	0
BT Lid Shaft	9	239	0
BT Lid Spring	10	267	0
Battery Lid	11	328	0
Shooter and Power	12	0	1
Strap shaft	13	198	0
Top Frame	14	993	0
Flash	15	0	1
Capacitor	16	0	1
Flash PCB	17	0	1
Main PCB	18	496	0
Support Jack	19	67	0
CCD	20	0	1
Filters	21	167	0
Barrier Ring	22	118	0
Ornamental Ring	23	0	1
Lens Sub-ASM	24	0	1
Lid	25	200	0
Jack Lid	26	118	0
Main Body	27	3088	0

DSC-T30 Component		IM(3)	Y(3)
Lens Cover	1	112	0
Cabinet Front	2	503	0
LCD Plate	3	350	0
LCD	4	350	0
LCD PCB	5	260	0
Buttons	6	0	1
Control Switch	7	0	1
LCD Window	8	188	0
Rear Cabinet	9	461	0
Insulating sheet	10	188	0
CCD Holder	11	178	0
CCD	12	0	1
Filters	13	167	0
Lens Sub-ASM	14	0	1
Plate	15	108	0
Flex PCB	16	488	0
Speaker	17	0	1
Main PCB	18	696	0
Flash	19	0	1
Capacitor	20	0	1
Flash PCB	21	0	1
Battery Lid	22	112	0
Main Body	23	1432	0

Figure 7. Impact metric (IM) vectors and decision vectors (Y) for the digital cameras.

below the threshold, defined at 50% for this case (see Section 4.3 for more details). The decision vectors are returned with the components to share from the cameras marked as ones (1) and the components to maintain unique for each product as zeros (0) (see Figure 7).

The decision vector for each product is then fed to the optimization algorithm (individual for each product) in which the dual objective function is used to find the optimum arrangement of components in clusters for the product. The chromosome for the optimum objective function value is returned and contains information about the component arrangement in clusters.

The chromosome contains the clustering strategy which includes clusters of components that are tightly related and single element clusters which are related to the components to be shared, since it is a premise of the analysis that a component would be easier to share if it is not part of any cluster unless the entire module is going to be shared. The chromosome may contain empty clusters indicating that a fewer number than the maximum of clusters is needed in order to achieve the optimum objective function value.

The results from the optimization (new component arrangement) alter the values of the IM. Therefore, there is the need to go back, calculate the new IM and rerun the process until there is no change in the decision vector in any of the products considered in the analysis. For the case of the three digital cameras hereby considered, only two additional iterations were necessary and Figure 7 shows a summary of the IM and Y (decision) vectors for the three cameras.

At the end of the process, the optimization algorithm returns the optimal chromosomes for the three cameras, representing the optimum component arrangement that not only preserves clusters

in which the components are tightly coupled, but also facilitates component sharing of the selected elements. The rearranged DSMs can be seen in Appendix A.

4.3. *Threshold level selection*

Threshold level is a key factor in setting the desired level of commonality among the products considered in the analysis. The threshold value is calculated as a percentage of the range of the IM values for each product; higher values of IM indicate that the component is not easy to change under the current architecture. The higher the threshold, the greater the number of components that can be considered as suitable candidates for sharing; therefore, it depends on the strategy of the company as to where this level should be set.

Nevertheless, for the purpose of this analysis, the parameter was varied until all the components would become candidates for sharing. The variation of the threshold value went from zero to 5.5 or 4.6, depending on the camera, at which level all the components were candidates for sharing. This variation served well for analysing the effect of the threshold level on the component sharing decision for these three products (see Figure 8).

The threshold level was set at 0.5 (*i.e.* 50% of the range of the IM score per product). At 0.5 some desired characteristics were observed: the components to share have very low IM scores (below 50% of the range); the number of components to share is not too high; and from that point on, big increments are required in order to get other components to be candidates.

On the other hand, the fact that the threshold needs to surpass four or five times the range of IM to make *all* the components become candidates for sharing (see the highest threshold value for each camera in Figure 8) can be understood when the functional matching process is examined. If multiple components are needed in order to match one or multiple functions, it is necessary to add the IM scores for those components; thus it is easy to go above the range.

One important clarification is that, for this case, the threshold was set constant among the three cameras; however, this is not a requirement for the framework application. Individual threshold values can be set for the different products achieving more commonality with a given variant rather than another.

	Threshold	Component																																
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
DSC-W100	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.2	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0.4	1	0	0	1	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0.5	1	0	0	1	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
	0.8	1	0	0	1	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
	1.1	1	0	0	1	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0
	2	1	0	0	1	0	0	0	1	0	0	0	0	1	1	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	1	0	1	1
	2.1	1	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1
	5.5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
DSC-S730	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
	0.2	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
	0.4	1	0	0	1	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0							
	0.5	1	0	0	1	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0							
	0.7	1	0	0	1	1	0	0	0	0	0	0	1	0	0	1	1	1	0	0	1	0	0	1	1	0	0	0							
	2.2	1	0	0	1	1	0	0	0	0	0	0	1	0	0	1	1	1	1	0	1	1	0	1	1	0	0	0							
	2.4	1	0	0	1	1	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1							
	4.6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1							
DSC-T30	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0											
	0.3	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0											
	0.4	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0											
	0.5	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0											
	0.7	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	1	0	1	1	1	0	0											
	2	0	0	0	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	1	1	1	1	1											
	2.1	0	0	0	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1											
	4.6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1											

Figure 8. Effect of threshold variation on component sharing.

5. Results and discussion

The results obtained by the framework application can be compared to the application of the algorithm proposed by Yu *et al.* (2007) in which there is no consideration regarding component sharing. A summary of the results can be seen in Appendix A.

It can be observed that the clustering strategies are quite different since the decision to share components modifies significantly the clusters that can be formed. For example, in the DSC-W100 the optimum without considering component sharing did not include a cluster with components *Capacitor Holder* and *Capacitor* (components 19 and 21, respectively), while in the optimum considering component sharing those components appear in a cluster (see Figure A3). Similarly, clusters that contained components selected for sharing changed, either reducing the number of components in the cluster, including additional components, or simply breaking down. These new structures for the products represent the optimal component arrangement in order to facilitate the sharing of the selected components according to the inherent connectivity among them, an aspect that has not been considered in this manner before.

6. Conclusions

This article introduced a framework to select the components to share in an automatic manner, among multiple products based on the architectural information contained in the product DSM and the Function–Component Matrix (FCM). It also proposes the optimization of the component arrangement in each product, finding the clusters that facilitate the component sharing while maintaining the compact modules in the product.

The process of selecting the candidates was based on the IM score of the components and the functional matching of the components across different products. The first indicated the impact of changing a component according to the current structure of the product. Functional matching was the means to identify components that have the same purpose in different products and may be suitable candidates for sharing.

In order to determine the optimal clustering while considering component sharing, both a product level representation (MDL) and a component level score (IM) were required and served as the objective function during the optimization of each product architecture.

Nevertheless, it is well known that product sharing decisions involve many different criteria (material affinity, serviceability, production costs, etc.) that have been ignored in the current framework. Therefore, it is necessary to continue to work along these lines in order to include some of those factors in facilitating the decision process.

Acknowledgements

This material is based on work supported by the National Science Foundation under Award No. 0726934. Any opinions, findings and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Browning, T.R., 2001. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Transactions on Engineering Management*, 48 (3), 292–306.

Browning, T.R. and Eppinger, S.D., 2002. Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE Transactions on Engineering Management*, 49 (4), 428–442.

Dahmus, J.B. and Gutowski, T.G., 2007. What gets recycled: an information theory based model for product recycling. *Environmental Science and Technology*, 41 (21), 7543–7550.

Eppinger, S.D., *et al.*, 1994. A model-based method for organizing tasks in product development. *Research in Engineering Design*, 6 (1), 1–13.
Gershenson, J.K., Prasad, G.J., and Zhang, Y., 2003. Product modularity: definitions and benefits. *Journal of Engineering Design*, 14 (3), 295–313.
Gershenson, J.K., Prasad, G.J., and Zhang, Y., 2004. Product modularity: measures and design methods. *Journal of Engineering Design*, 15 (1), 33–51.
Khajavirad, A. and Michalek, J., 2008. A decomposed gradient-based approach for generalized platform selection and variant design in product family optimization. *Journal of Mechanical Design*, 130 (7), 071101 (8pp).
Khajavirad, A., Michalek, J., and Simpson, T., 2009. An efficient decomposed multiobjective genetic algorithm for solving the joint product platform selection and product family design problem with generalized commonality. *Structural and Multidisciplinary Optimization*, 39 (2), 187–201.
Kwak, M., Cho, N.W., and Hong, Y.S., 2007. Eco-architecture analysis as a method of end-of-life decision making for sustainable product design. *In*: *Proceedings of the ASME 2007 international design engineering technical conference*, 4–7 September 2007, Las Vegas, NV. Vol. 3A. New York: ASME.
Martin, M.V., 1999. Design for variety: a methodology for developing product platform architectures. Thesis (PhD). Stanford University, Stanford, CA.
Martin, M. and Ishii, K., 2002. Design for variety: developing standardized and modularized product platform architectures. *Research in Engineering Design*, 13 (4), 213–235.
Martin, M. and Ishii, K., 1996. Design for variety: a methodology for understanding the costs of product proliferation. *In*: *Proceedings of the ASME 1996 design engineering technical conference*, 18–22 August 1996, Irvine, CA. New York: ASME.
Meyer, M.H. and Lehnerd, A.P., 1997. *The power of product platform: building value and cost leadership*. New York: Free Press.
Mikkola, J.H. and Gassmann, O., 2003. Managing modularity of product architectures: toward an integrated theory. *IEEE Transactions on Engineering Management*, 50 (2), 204–218.
Moon, S.K., Simpson, T.W., and Kumara, S.R., 2008. A strategic module-based platform design method for developing customized products in dynamic and uncertain market environments. *In*: *Proceedings of the ASME 2008 international design engineering technical conference*, 3–6 August, Brooklyn, NY. Vol. 4. New York: ASME.
Pandey, V. and Thurston, D., 2008. Metric for disassembly and reuse: formulation and validation. *In*: *Proceedings of the ASME 2008 international design engineering technical conference*, 3–6 August, Brooklyn, NY. Vol. 4. New York: ASME.
Pimmler, T.U. and Eppinger, S.D., 1994. Integration analysis of product decompositions. *In*: *Proceedings of the ASME 6th international conference on design theory and methodology*, 11–14 September, Minneapolis, MN. Vol. 68. New York: ASME, 343–351.
Rojas, A.J. and Esterman, M., 2008. A measure of impact for platform changes. *In*: *Proceedings of the ASME 2008 international design engineering technical conference*, 3–6 August, Brooklyn, NY. Vol. 4. New York: ASME.
Rojas-Arciniegas, A.J., 2008. A selection framework for derivative products: development of an impact metric and platform value assessment methodology. Thesis (Master's). Department of Industrial Engineering, Rochester Institute of Technology, Rochester, NY.
Simpson, T.W., 2004. Product platform design and customization: status and promise. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 18 (1), 3–20.
Smaling, R. and de Weck, O., 2007. Assessing risks and opportunities of technology infusion in system design. *System Engineering*, 10 (1), 1–25.
Sony, 2006a. DSC-T30 Service Manual – Level 2, ver. 1.0, DI Technical Support Department, Sony EMCS Co., Aichi, Japan.
Sony, 2006b. DSC-W100 Service Manual – Level 2, ver. 1.0, DI Technical Support Department, Sony EMCS Co., Aichi, Japan.
Sony, 2007. DSC-S730 Service Manual, ver. 1.0, Kohda TEC, Sony EMCS Co., Aichi, Japan.
Sosa, M.E., Eppinger, S.D., and Rowles, C.M., 2007. A network approach to define modularity of components in complex products. *Journal of Mechanical Design*, 129 (11), 1118–1129.
Steva, E.D., *et al.*, 2006. Two methodologies for identifying product platform elements within an existing set of products. *In*: *Proceedings of the ASME 2006 international design engineering technical conference*, 10–13 September 2006, Philadelphia, PA. Vol. 4. New York: ASME, 811–821.
Steward, D., 1981. The design structure system: a method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, 28 (3), 71–74.
Strawbridge, B., McAdams, D., and Stone, R.B., 2002. A computational approach to conceptual design. *In*: *Proceedings of the ASME 2002 design engineering technical conference*, 2002, Montréal, Quebec, Canada. Vol. 4. New York: ASME, 15–25.
Thevenot, H.J. and Simpson, T.W., 2006. Commonality indices for product family design: a detailed comparison. *Journal of Engineering Design*, 17 (2), 99–119.
Tucker, C.S. and Kim, H.M., 2008. Optimal product portfolio formulation by merging predictive data mining with multilevel optimization. *Journal of Mechanical Design*, 130 (4), 041103 (15pp).
Ulrich, K.T., 1995. The role of product architecture in the manufacturing firm. *Research Policy*, 24 (3), 419–440.
Ulrich, K.T. and Eppinger, S.D., 2004. *Product design and development*. 3rd edn. New York: McGraw-Hill.
Wang, B., 2007. Information-theoretic methods for modularity in engineering design. Thesis (PhD). California Institute of Technology, Pasadena, CA.

Wang, B. and Antonsson, E.K., 2004. Information measure for modularity in engineering design. *In*: *Proceedings of the ASME 2004 international design engineering technical conference*, 28 September–2 October 2004, Salt Lake City, UT. Vol. 3. New York: ASME, 449–458.

Wang, B. and Antonsson, E.K., 2005. Hierarchical modularity: decomposition of function structures with the minimal description length principle. *In*: *Proceedings of the ASME 2005 international design engineering technical conference*, 24–28 September 2005, Long Beach, CA. Vol. 5. Brooklyn, NY: ASME, 393–402.

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

Zacharias, N.A. and Yassine, A.A., 2007. Platform investment decisions in product family design. *In*: *Proceeding of the ASME 2007 international design engineering technical conference*, 4–7 September 2007, Las Vegas, NV. Vol. 3A. Brooklyn, NY: ASME.

Appendix A. Optimum clustering and component sharing candidates for digital cameras

DSC-T30	Component #	Buttons	Control Switch	CCD	Lens Sub-ASM	Speaker	Flash	Capacitor	Flash PCB	LCD Plate	LCD	LCD PCB	Rear Cabinet	Cabinet Front	Main Body	Plate	Flex PCB	Main PCB	CCD Holder	Filters	Lens Cover	LCD Window	Insulating sheet	Battery Lid
Component #		6	7	12	14	17	19	20	21	3	4	5	9	2	23	15	16	18	11	13	1	8	10	22
Buttons	6	0	0	0	0	0	0	0	0	0	0	15	9	0	0	0	0	0	0	0	0	0	0	0
Control Switch	7	0	0	0	0	0	0	0	0	0	0	15	9	0	0	0	0	0	0	0	0	0	0	0
CCD	12	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	9	9	9	0	0	0	0
Lens Sub-ASM	14	0	0	18	0	0	0	0	0	0	0	0	0	15	9	0	0	9	3	6	0	0	0	0
Speaker	17	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	6	6	0	0	0	0	0	0
Flash	19	0	0	0	0	0	0	9	9	0	0	0	0	9	9	0	0	0	0	0	0	0	0	0
Capacitor	20	0	0	0	0	0	9	0	6	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0
Flash PCB	21	0	0	0	0	0	9	6	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0
LCD Plate	3	0	0	0	0	0	0	0	0	0	24	9	0	0	9	0	0	0	0	0	0	0	0	0
LCD	4	0	0	0	0	0	0	0	0	24	0	9	0	0	0	0	0	0	0	0	0	9	0	0
LCD PCB	5	15	15	0	0	0	0	0	0	9	9	0	3	0	0	0	0	9	0	0	0	0	0	0
Rear Cabinet	9	9	9	0	0	0	0	0	0	0	0	3	0	18	18	0	0	0	0	0	0	9	0	0
Cabinet Front	2	0	0	0	15	0	9	0	0	0	0	0	18	0	18	0	0	0	0	0	9	0	3	0
Main Body	23	0	0	0	9	9	9	9	0	9	0	0	18	18	0	6	6	18	6	0	0	0	0	9
Plate	15	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	9	12	0	0	0	0	0	0
Flex PCB	16	0	0	0	0	6	0	0	0	0	0	0	0	0	6	9	0	21	0	0	0	0	0	0
Main PCB	18	0	0	9	9	6	0	0	9	0	0	9	0	0	18	12	21	0	0	0	0	0	15	0
CCD Holder	11	0	0	9	3	0	0	0	0	0	0	0	0	0	6	0	0	0	0	3	0	0	0	0
Filters	13	0	0	9	6	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
Lens Cover	1	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0
LCD Window	8	0	0	0	0	0	0	0	0	0	9	0	9	0	0	0	0	0	0	0	0	0	0	0
Insulating sheet	10	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	15	0	0	0	0	0	0
Battery Lid	22	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0

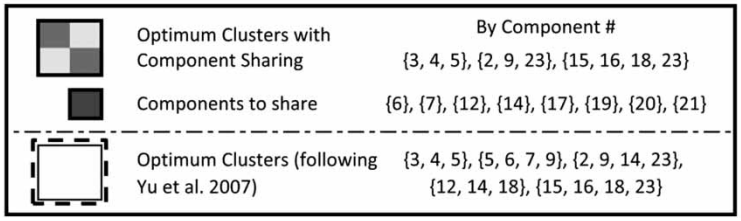

Figure A1. Optimum DSM for the Sony digital camera DSC-T30.

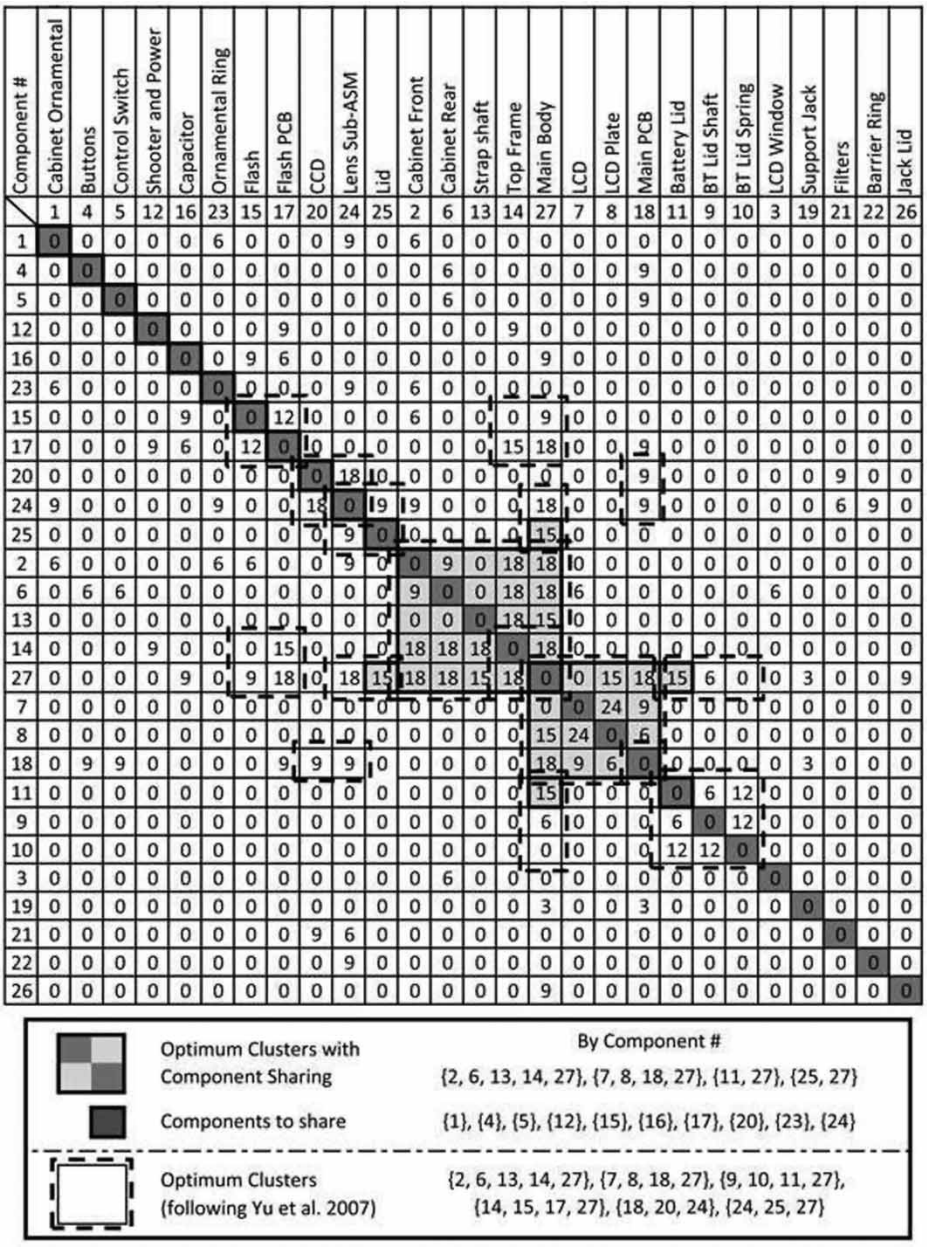

Figure A2. Optimum DSM for the Sony digital camera DSC-S730.

Component #		1	4	8	13	14	16	20	23	26	9	10	11	19	21	30	32	33	27	2	6	34	28	12	15	17	18
Ornamental Ring	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	6	0	0	0	0
Buttons	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0
Control Switch	8	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	6	0	0	0	0	0	0
Shoot Button	13	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	6	0
Zoom Slider	14	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	6	0
Speaker	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	1	0	0	0	0	3	0
Flash	20	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	6	0	3	3	0	0	0	0
CCD	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	18	0	0	0	0
Ornamental Ring	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0
LCD	9	0	0	0	0	0	0	0	0	0	0	24	9	0	0	0	0	0	0	0	6	0	0	0	0	0	0
LCD Plate	10	0	0	0	0	0	0	0	0	0	24	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LCD PCB	11	0	0	6	0	0	0	0	0	0	9	6	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0
Capacitor Holder	19	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0
Capacitor	21	0	0	0	0	0	0	9	0	0	0	0	0	6	0	0	0	0	0	0	0	0	3	0	0	0	0
Top Flex PCB	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	9	0	0	0	3	0	0	0	0	0
Front PCB	32	0	0	0	3	3	3	0	9	0	0	0	9	0	0	9	0	15	0	0	0	0	9	0	3	0	0
Battery Holder	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	15	0	0	0	0	9	0	6	0	0	0
Visor	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	6	0	9	0	0	0	0
Cabinet Front	2	6	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	6	0	15	18	9	1	1	0	9
Cabinet Rear	6	0	6	6	0	0	1	0	0	0	6	0	0	0	0	0	0	0	6	15	0	18	0	1	0	0	9
Main Body	34	0	0	0	0	0	0	3	0	0	0	0	0	0	0	3	0	9	0	18	18	0	18	6	0	0	0
Lens Sub-ASM	28	6	0	0	0	0	0	3	18	9	0	0	0	0	3	0	9	0	9	9	0	18	0	0	0	0	0
Side Frame 1	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	1	1	6	0	0	0	3	0
Mic	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	1	0	0	0	0	0	6	0
Top Frame	17	0	0	0	6	6	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	6	0	6
Side Frame 2	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	9	0	0	0	0	6	0
LCD Window	3	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0
Side Lid	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0
Side Cover	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	0	0	0	0	0	0
Flash PCB	22	0	0	0	0	0	0	12	0	0	0	0	0	0	6	0	9	0	0	0	0	0	6	0	0	0	0
Filters	24	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0
Barrier Ring	25	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0
Rear PCB	29	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	12	0	0	0	0	0
Lid	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0

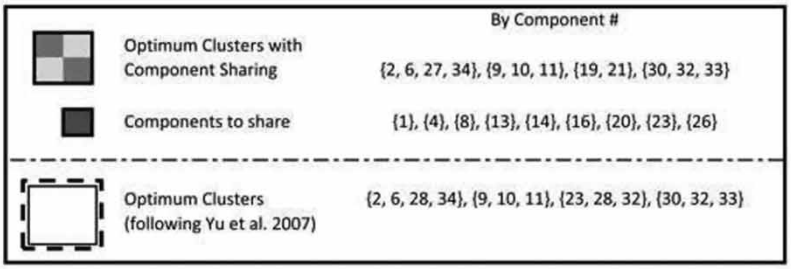

Figure A3. Optimum DSM for the Sony digital camera DSC-W100. The last eight columns are excluded since the DSM is symmetric and all the elements in the diagonal block are zero.