Ordering Cross-Functional Decision Making in Product Development

Viswanathan Krishnan
Steven D. Eppinger
Daniel E. Whitney

Revised October 1992
WP # 3299-91 MS

Key Words: Cross-Functional Product Development, Design Quality.

Send Correspondences to:
Prof. Steven D. Eppinger
MIT Sloan School of Management
30, Wadsworth Street, E53-347
Cambridge, MA 02139
Ph: (617) 253-0468 email: eppinger@eagle.mit.edu

Ordering Cross-Functional Decision Making in Product Development

Viswanathan Krishnan

Steven D. Eppinger

Daniel E. Whitney

Massachusetts Institute of Technology

Cambridge, MA 02139

Abstract

In order to develop implementation strategies for concurrent engineering, we study the problem of decoupling decisions in a complex design project. Since true simultaneous engineering involves high coordination costs, inherently coupled tasks must often be executed sequentially. Furthermore, to keep on schedule, we assume that no backtracking is allowed. This paper explores the ordering of such tasks in search of near-optimal solutions without iteration. To formulate this problem, we introduce the notion of *Quality Loss* to quantify the loss of design freedom incurred by the decision makers in the downstream stages of a cross-functional team. We then define the optimal sequence as the decision order with the lowest quality loss. To do this efficiently, we first partition the design variables into *exclusive groups* based on their connectivities. Next, the sensitivities and connectivities are combined to characterize several types of quality-invariant decisions. Sufficient conditions, relating exclusive group structure to invariant decisions, are presented to reduce the complexity of identifying the optimal order. These ideas are illustrated using a DC motor design example.

1. Cross-Functional Decision Making in Product Development

A *cross-functional product development team* brings together decision makers involved with multiple product life-cycle issues: function, geometry, manufacturing, maintenance etc. In recent years, many product development organizations have been restructured into cross-functional teams, reflecting their resolve to improve product quality along multiple dimensions while reducing the development lead time. However, as Clark and Fujimoto observe[2], use of cross-functional teams alone does not guarantee effective development; the teams also need to be tightly integrated and strongly coordinated. Simulations conducted by Gebala and by Pekar[5, 6] show that a weakly coordinated cross-functional team can perform worse than functionally integrated teams. This is because confusions arise from conflicting goals and cause iterations, delays, and rework of the design tasks. Field studies conducted by Roodvoets at a U.S. automaker[7] also show that the

change to a cross-functional team structure without proper understanding may result in an increase in the development lead time.

A significant challenge facing cross-functional product development is the lack of understanding of the team-based decision process. Since the decisions made by the various functional decision makers are generally coupled and often in conflict, the absence of a proper decision strategy for cross-functional teams can lead to poor designs and/or unnecessary iteration. We have been studying how industrial groups perform product development and we have found that most firms do not understand the fundamental coupling underlying their design problems. We believe that the structure of this inherent coupling determines the decision strategy which must be employed.

Naturally uncoupled problems, in which all decisions (variables) are independent, are quite rare, and we have found none in our industrial field work. Rather, we find problems in which the design decisions interact in interesting ways that are not usually understood by any one person on the project. To assist, we are developing design representation and analysis tools to identify the relationships among the various design tasks [4]. When decisions are tightly coupled, designers have two choices: either they attempt to address the coupled issues simultaneously by negotiation or iteration, or they can find ways to tackle the issues sequentially by "optimally" eliminating the task coupling. In work with Smith, we have addressed the analysis of iterative design procedures [9]. In this paper, we formulate the strategy of ordering cross-functional decision making that helps to eliminate the need for iteration.

2. Ordering the Cross-Functional Decision Making Process

We model the cross-functional product design team as involving n designers, each entrusted with a particular product function[1]. We further assume that the individual product functions are parametric design activities (e.g. sizing), not conceptual design (e. g. shape design) or detailed design activities (e.g. storing and distributing drawings).

By *(cross-functional) decision order*, we mean the decision process in which each member of the cross-functional team makes necessary decisions exactly once, in a predetermined sequence. Although subsequent negotiation among the cross-functional team members may be needed in improving the solution, ordering the team members for decision making is useful because frequently the products can either be too complex or time-critical to indulge in negotiations.

---

[1]We distinguish a cross-functional product development team in which the individual decision makers specialize in particular disciplines, from an ideal multidisciplinary development team in which the designers are trained in multiple disciplines.

This effect is illustrated by our study of the design procedures in two competing firms developing a prototypical product (an electro-mechanical instrument). As shown in Figure 1, the designers in one firm recognize three aspects of the product design (casing, wiring, and optics) to be so tightly coupled that they must be designed simultaneously, requiring lengthy negotiation (five to ten design iterations, taking up to six months) before enough detail can be settled to build the first working prototype. The designers in the competing firm believe that a first prototype must be delivered much more quickly and adopt a more sequential decision order where the wiring design is left for last. The design is completed faster (in just a few weeks), and the prototype is built with rather crude wiring, which is revised later for the second prototype.

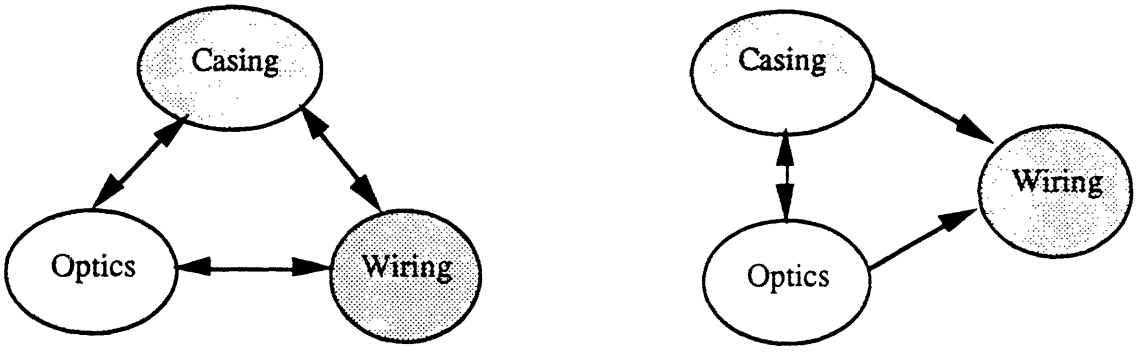

Figure 1. Concurrent and Sequential Decision Strategies

In this case, the sequential decision strategy greatly accelerated the project, despite the slight loss in quality (poor wiring for the first prototype). It was wise to schedule the wiring design as the last task, since the quality is affected in such a minor way. However in general, there exist multiple decision orders by which the members of such a cross-functional team may make decisions and in the absence of negotiation, the different orders differ substantially in the quality of the designs that they produce. The primary difference among the different orders is in the degrees of freedom decided in the upstream stages of the decision process. *Decision orders that result in a good quality product typically have the upstream decisions not causing too much of a quality loss for the subsequent cross-functional decision makers.* In bad decision orders, the decision-making capability of the team members in the downstream stages is severely impaired due to the *order constraints* imposed by the prior decision makers. For example, in a camera design project, a decision to save cost by integrating several parts of a complex assembly may lead to increased complexity, lead time, and cost in molding the integrated component. It is important to quantify the loss in quality suffered by downstream decision makers due to the decisions made in the upstream stages.

In the following section we introduce the notion of *quality loss* to capture the loss in quality of the subsequent decision makers of a cross-functional team. In later sections, we will study the decision order that involves the least quality loss for the design process.

3.1 Terminology

For this analysis, we use the following definitions, illustrated in the next subsection.

• P is a cross-functional product development process with n decision makers, entrusted with executing n cross-functional decision making tasks $T_1, T_2, T_3, \ldots, T_n$ and thereby choosing the values of m parametric design variables $x_1, x_2, x_3, \ldots, x_m$. Let X be a set comprising of all the design variables $x_1, x_2, x_3, \ldots, x_m$.

• Each task or function T_i, upon execution produces a functional output J_i . Let $\mathbf{Z}_i$ be the set (and z_i a vector) consisting of all the design variables that could possibly be decided by T_i. ($\mathbf{Z}_i \subset X$). The variables belonging to $\mathbf{Z}_i$, $\{z_{i1}, z_{i2}, \cdots, z_{ir}\}$, are said to occur in task T_i .

• The decision process in which the task T_i is not subject to any order constraints is called the independent decision process. Let J_i^* represent the functional output of task T_i in the independent decision process; the value set by T_i for the design variables in the independent decision process will be called independent decisions.

• A *decision order* φ is a sequence of the n tasks, $T_1, T_2, T_3, \ldots, T_n$, such that:

1) The tasks in the design process are executed in the order in which they occur in φ .

2) Each task T_i upon execution decides on a value for all undecided variables belonging to $\mathbf{Z}_i$. The output of task T_i in the order φ will be denoted by J_i^{φ}. Let $\overline{\mathbf{Z}_i}$ denote the complement of $\mathbf{Z}_i$. Consider the order $\varphi = \{T_1, T_2, T_3, \ldots, T_n\}$. In this order, T_i decides the values of all variables in $\mathbf{Z}_i \cap \overline{\{\mathbf{Z}_1 \cup \mathbf{Z}_2 \cup \cdots \cup \mathbf{Z}_{i-1}\}}$. It faces order constraints of the form, $z_{iq} - z_{iq}^{\varphi} = 0$ for $z_{iq} \in \mathbf{Z}_i \cap \{\mathbf{Z}_1 \cup \cdots \cup \mathbf{Z}_{i-1}\}$. In other words, T_i loses freedom in $\mathbf{Z}_i \cap \{\mathbf{Z}_1 \cup \mathbf{Z}_2 \cup \cdots \cup \mathbf{Z}_{i-1}\}$.

• A design variable x_k's value in the ordered design process is *decided* by the first task in the order in which it occurs.

• A task T_i is said to lose the design freedom x_k in the order φ, if $x_k \in \mathbf{Z}_i$ and the value of x_k in the order φ is decided by a preceding task.

Defn: Quality Loss incurred by task T_i in a decision order φ, QL_i^{φ}, is defined to be the nonnegative offset of the output of task T_i in φ, J_i^{φ}, from J_i^*

$$QL_i^{\varphi} = \left| J_i^{\varphi} - J_i^{*} \right|$$

Defn: Quality Loss of an order φ, QL^{φ}, is defined to be the weighted sum[2] of quality losses incurred by each task T_i in φ.

$$QL^{\varphi} = \sum_{i=1}^{n} w_i \, QL_i^{\varphi}$$

Defn: A decision order φ^* is defined to be the *optimal decision order* if its quality loss $QL^{\varphi^*} \leq QL^{\varphi}$ for all possible φ.

3.2 Framework

It would be difficult to understand the characteristics of cross-functional decision making in the absence of any specific interpretations for design tasks. To focus on the problem, an individual team member's task is interpreted as the optimization of a particular design criterion. This provides a unified basis to model the behavior of the multiple decision makers in cross-functional product design. Although in routine life, people may be satisficers rather than optimizers[8], in many commercially competitive activities product developers are required to obtain optimal results. In the design of complex and novel technologies such as hypersonic aircrafts, using the optimal results may make the difference between "flying and staying on the ground"[10]. In this paper, we further assume every functional design task has the form of a nonlinear program, as is the case with several parametric design activities. In the next few subsections we relate the optimal order (for execution of the cross-functional decision tasks) to the underlying structure of the design interactions, illustrated through the example of the design of a dc (direct current) motor.

3.3 Design Problem

To decide on the order in which decisions should be made by the members of a cross-functional team, whose members are concerned respectively with maximizing the torque generated by the DC motor (performance), minimizing the area occupied by the stator (size) and minimizing the cost of materials (sum of the area occupied by the steel portion of the rotor and area of copper). The variables to be decided are given in Table 1.

The parametric design relations are shown in Table 2 (formulated using design constraints from [3]. All design equality constraints have been explicitly eliminated or used in symbolic propagation. The only inequalities given are lower and upper limit constraints). We will assume that each team member operates with his/her own model of the design and objectives. Table 3

[2]It is noteworthy that Quality Loss of an order is defined here as a linear function of the individual quality losses. We plan to investigate other definitions in future work.

shows the results if the designers were to make decisions independently; the outputs under these circumstances being the independent outputs and the decisions, independent decisions.

Decision Variable	Symbol	Bounds
Armature diameter	ad	$10 \le ad \le 12$ (inches)
Motor inner diameter	id	$0.1 \le id \le 3.0$ (inches)
Motor outer diameter	od	$20 \le od \le 24$ (inches)
Diameter of windings	dw	$0.01 \le dw \le 0.2$ (inches)
Current density	cd	$0.1 \le cd \le 50.0$ (amp / in^2)
No. of armature windings	nw	$1 \le nw \le 1500$ (turns)
Thickness of magnet used	tm	$0.05 \le tm \le 1.0$ (inches)

Table 1. Design Variables for a DC motor

Task	Task Description	Analytical Forms (Minimizations)
T_1	Maximize Torque Generated	$J_1 = -1.57\ cd\ dw^2$
T_2	Minimize Space	$J_2 = 0.785\ (od^2 - ad^2) - 0.26\ nw\ dw^2$
T_3	Minimize Material Costs	$J_3 = 0.785\ (ad^2 - id^2) - 2.1\ ad\ tm + 0.785\ dw^2$

Table 2. Objectives as Functions of Variables

Task	Independent Decisions and Outputs
T_1	$dw_1^* = 0.2;\ cd_1^* = 50;\ J_1^* = -3.14$
T_2	$ad_2^* = 12;\ od_2^* = 20;\ dw_2^* = 0.2;\ nw_2^* = 1500;\ J_2^* = 185.3$
T_3	$ad_3^* = 10;\ id_3^* = 3;\ dw_3^* = 0.01;\ tm_3^* = 1.0;\ J_3^* = 50.4$

Table 3. Independent Decisions and Outputs

One observes that designers executing tasks T_1 and T_2 independently drive the variable dw to its upper limit, while the designer entrusted with T_3 drives dw to its lower limit. When the designers make their decisions in a particular order, only one of them can decide the value of dw; subsequent decision makers are constrained by this value of the variable dw. An example of a

decision order is given below where the tasks are weighted[3] in the inverse ratio of the magnitude of their independent solutions, i. e. $w_i = 1/J_i^*$.

The order $\{T_1, T_3, T_2\}$ produces the following results:

Stage 1 Minimize $J_1 = -1.57\ cd\ dw^2$

Decision: $cd = 50$; $dw = 0.2$; $J_1 = -3.14$; $w_1 QL_1 = 0.0$

Stage 2 Minimize $J_3 = 0.785\ (ad^2 - id^2) - 2.1\ ad\ tm + 1.57\ dw^2$
subject to $dw = 0.2$;

Decision: $ad = 10$; $id = 3$; $tm = 1$; $J_3 = 50.5$; $w_3 QL_3 = 0.0012$

Stage 3 Minimize $J_2 = 0.785\ (od^2 - ad^2) - 0.26\ nw\ dw^2$
subject to $ad = 10$; $dw = 0.2$;

Decision: $od = 20$; $nw = 1500$; $J_2 = 219.9$; $w_2 QL_2 = 0.188$

Total $QL = w_1 QL_1 + w_2 QL_2 + w_3 QL_3 = 0.189$

The straightforward method to determine the optimal order[4], by explicitly considering all orders and evaluating their quality losses, is shown in Table 4. Evaluating the quality loss of each decision order requires the execution of every task because their results depend on their position in the decision order. So in this case a total of 3! x 3 = 18 nonlinear programs need to be solved to identify that $\{T_1, T_3, T_2\}$ is the optimal decision order with the lowest quality loss. In large designs, determining the optimal order becomes tedious requiring $Order(n!\ n)$ optimizations. Evidently, even if the tasks may not be nonlinear programs, exhaustive enumeration is prohibitively complex.

Inspection of Table 4 shows that the values of several design decisions (for example, *cd*) are invariant from one order to another while some design variable values do not vary over a subset of all orders (for example the value of *ad* is the same in three of the orders). These invariances can be systematically utilized to relate the quality loss of a decision order to the design structure. Quality loss of the individual tasks varies with the decision order because different degrees of freedom are lost in different decision orders. However there exists a subset of decision orders over which a particular task loses a particular design freedom. For instance, in four of the six orders T_3 loses

---

[3] Choosing weights for the design tasks is an issue not addressed in this paper. In the DC motor problem we scale the different tasks in the inverse ratio of their independent decisions so as to normalize all quality losses; the methods developed in this paper can accommodate any choice of weights.

[4] It is noteworthy that the multiobjective optimal solution that minimizes the total quality loss does not correspond to a realistic cross-functional decision process solution in which knowledge of the design functions is localized.

the design freedom, *dw*. In all these orders the design variable assumes the same value, *dw* = 0.2. and T_3 faces the same order constraint, perhaps resulting in the same quality loss. How could we exploit invariant design variable values which translate into invariant order constraints? If we can decompose the quality loss incurred by a task into components due to loss of individual design degrees of freedom, then we can compute the order variant quality loss from order invariant terms. Then the following questions arise: What is the general structure in such problems that leads to order invariance? Can the quality loss of every order be decomposed into a certain small number of order invariant quantities, calculated *a priori*?

Order	Design Decisions made during the order	QL
$\{T_1, T_2, T_3\}$	*ad*=12; id=3; *od* = 20; *dw*=0.2; *cd*= 50; *nw*=1500; *tm* =1.0	0.603
$\{T_1, T_3, T_2\}$	*ad*=10; id=3; *od* = 20; *dw*=0.2; *cd*=50; *nw*=1500; *tm* =1.0	0.189
$\{T_2, T_1, T_3\}$	*ad*=12; id=3; *od* = 20; *dw*=0.2; *cd*=50; *nw*=1500; *tm* =1.0	0.603
$\{T_2, T_3, T_1\}$	*ad*=12; id=3; *od* = 20; *dw*=0.2; *cd*=50; *nw*=1500; *tm* =1.0	0.603
$\{T_3, T_1, T_2\}$	*ad*=10; id=3; *od* = 20; *dw*=0.01; *cd*=50; *nw*=1500; *tm* =1.0	1.268
$\{T_3, T_2, T_1\}$	*ad*=10; id=3; *od* = 20; *dw*=0.01; *cd*=50; *nw*=1500; *tm* =1.0	1.268

Table 4. Results of All Decision Orders

3.4 Exclusive Groups

It is interesting to observe that in ordering cross-functional decision making, if two design variables x and y occur in the same combination of design tasks, then the decision about the values of both x and y will *always* be made by the same team person. Any subsequent member who loses the design freedom x will also lose design freedom y to the same member that decided x. When we compute the quality loss due to loss of x, we will also compute the quality loss due to loss of y. So all variables that occur in the same combination of tasks can be grouped together as shown in Figure 2. The quality loss incurred by a task needs only to be decomposed into components over losses of such groups of design freedom. Any cross-functional team member possesses or loses design freedom in groups. In the three task DC motor problem, design variables get partitioned into seven groups. Because the combinations are exclusive to each other, the groups are called exclusive groups.

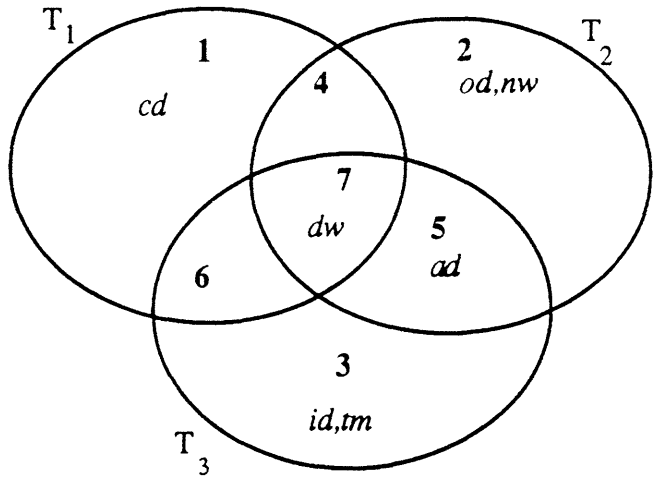

Figure 2. Exclusive Groups in the DC Motor Example

An exclusive group consists of variables that appear in the same combination of tasks. For a n task design process there are $M = 2^n$-1 groups, which correspond to the Boolean combination of the complements given below[5].

$$\begin{Bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_i \\ \vdots \\ Y_M \end{Bmatrix} = \begin{Bmatrix} (Z_1 \cap \overline{Z_2} \cap \overline{Z_3} \cap \cdots\cdots \cap \overline{Z_n}) \\ (\overline{Z_1} \cap Z_2 \cap \overline{Z_3} \cap \cdots\cdots \cap \overline{Z_n}) \\ \vdots \\ (Z_1 \cap Z_2 \cap \overline{Z_3} \cap \cdots\cdots \cap \overline{Z_n}) \\ \vdots \\ (Z_1 \cap Z_2 \cap Z_3 \cap \cdots\cdots \cap Z_n) \end{Bmatrix}$$

The tasks in which the variables belonging to any exclusive group Y_i occur will be referred to as tasks *forming* Y_i. For example, the tasks forming Y_7 are T_1, T_2 and T_3. The tasks themselves are said to be *spanned* by the various exclusive groups.

In Figure 2, T_3 is spanned by Y_3, Y_5, Y_6 and Y_7. Let $\mathbf{y}_j$ be the vector of variables belonging to Y_j, $\mathbf{t}_j$ be the set of tasks forming Y_j. For the DC motor problem, we have:

$$z_1 = \begin{Bmatrix} cd \\ dw \end{Bmatrix}; \quad z_2 = \begin{Bmatrix} ad \\ od \\ dw \\ nw \end{Bmatrix}; \quad z_3 = \begin{Bmatrix} ad \\ id \\ dw \\ tm \end{Bmatrix}$$

$$y_1 = \{cd\}; \; y_2 = \begin{Bmatrix} ad \\ nw \end{Bmatrix}; \; y_3 = \begin{Bmatrix} id \\ tm \end{Bmatrix}; \; y_4 = \{\varnothing\}; \; y_5 = \{ad\}; \; y_6 = \{\varnothing\}; \; y_7 = \{dw\}$$

[5]In a real product design problem, n is about 5, while m may be 1000; n << m, making the grouping of variables useful.

$t_1 = \{T_1\}$; $t_2 = \{T_2\}$; $t_3 = \{T_3\}$; $t_4 = \left\{ \begin{matrix} T_1 \\ T_2 \end{matrix} \right\}$; $t_5 = \left\{ \begin{matrix} T_2 \\ T_3 \end{matrix} \right\}$; $t_6 = \left\{ \begin{matrix} T_1 \\ T_3 \end{matrix} \right\}$; $t_7 = \left\{ \begin{matrix} T_1 \\ T_2 \\ T_3 \end{matrix} \right\}$

The decision made by a task T_j for the values of the variables in a particular exclusive group Y_i is referred to as the exclusive group Y_i decision by task T_j. The loss of design freedom in deciding the values of all variables belonging to an exclusive group is referred to as loss of exclusive group freedom. Notice that in the DC motor case, the values of all the design variables in certain exclusive groups, such as Y_3, do not vary from one order to another (Table 4). Such groups are called *order-invariant exclusive groups.*

Defn: An exclusive group Y_j is said to be order invariant if the value of each design variable belonging to Y_j is the same under every order.

Why are some exclusive groups not order invariant? There are two reasons why a certain exclusive group decision may vary from one order to another. *First,* different designers may decide the values of design variables in an exclusive group in different orders. For example, in the order $\{T_3, T_1, T_2\}$, the team member responsible for the first task (T_3) makes the exclusive group Y_7 decision and sets the value of variable *dw* to 0. 01, while in the order $\{T_1, T_3, T_2\}$, the designer responsible for T_1 makes the exclusive group Y_7 decision (sets *dw* to 0. 2). *Secondly*, even when the same task makes an exclusive group decision, the value decided may vary from one order to another, because the task may face different constraints in different orders (because of different predecessors). If however, the order constraints do not affect the exclusive group Y_i decision made by T_j, then the exclusive group Y_i decision by task T_j is the same as its independent decision. What this means is that the exclusive group Y_i is invariant over a subset of all orders, the subset in which it is decided by the forming task T_j.

Defn: An exclusive group Y_j is said to be *forming task T_i invariant* if the value decided by T_i for the design variables belonging to Y_j is the same in every order in which T_i makes Y_j decision.

Notice that if an exclusive group is order invariant, its value does not vary under any order, and so it is also forming task invariant. However, an exclusive group that is forming task invariant is not necessarily order invariant. The motive behind pursuing these two invariances is to use them to facilitate the identification of the optimal order. For instance, if all the exclusive groups spanning a certain design task are forming task invariant, then the designer entrusted with the task need not repeat the decision making process (after having identified the independent decisions). Identifying order invariance impacts (by reducing the size of) every task in the design process. The question arises as to how one may identify these two invariances. There are several situations under which exclusive groups may be order invariant or forming task invariant. Towards

characterizing these properties, we focus in our research on identifying increasingly stronger sufficient conditions. As a first step, we state the following propositions (proved in appendices) for design problems where all design objectives are continuously differentiable and explicitly expressible as functions of design variables (through serial constraint sets) and all inequality constraints are range constraints.

Proposition 1 An exclusive group Y_j is order invariant if C1 (C1.1 *or* C1.2) is satisfied.

C1.1 $\frac{\partial}{\partial y_k}\frac{\partial J_p}{\partial y_j} = 0 \; \forall \; y_k, (j \neq k)$ spanning every $T_p \in t_j$.

(i. e. the exclusive group is insensitive to other exclusive groups spanning every forming task) and the independent decisions of all forming tasks equal the same value y_j^*.

C1.2 In the range of the design problem in question, each design variable in Y_j is monotonic in every forming task (partial of the forming task with respect to the design variable is sign invariant in the range of the design problem); further, the monotonicity of a variable is of the same type (increasing or decreasing) in all tasks:

$$\forall \; X_i \in Y_k, \text{ either } \frac{\partial J_p}{\partial X_i} > 0 \; \forall T_p \in t_j \text{ or } \frac{\partial J_p}{\partial X_i} < 0 \; \forall T_p \in t_j$$

Proposition 2 Exclusive group Y_j is forming task T_i invariant if it satisfies C2.1 *or* C2.2

C2.1 $\frac{\partial}{\partial y_k}\frac{\partial J_i}{\partial y_j} = 0 \; \forall \; y_k, (j \neq k)$ spanning T_i and not satisfying C1.

(i. e. y_j decision by T_i is sensitive only to other provenly order invariant exclusive groups)

C2.2 $\forall \; X_i \in Y_k$, either $\frac{\partial J_p}{\partial X_i} > 0$ or $\frac{\partial J_p}{\partial X_i} < 0$ (i. e. every Variable in y_j is monotonic in T_i in the range of the given design problem).

$\frac{\partial J_3}{\partial y_3} = \begin{Bmatrix} -2.1\ ad \\ -1.57\ id \end{Bmatrix} < 0$
$\frac{\partial J_3}{\partial y_7} = \{1.57\ dw\} > 0$
$\frac{\partial J_3}{\partial y_5} = \{1.57\ ad - 2.1\ tm\} = \frac{\partial J_3}{\partial y_5}(y_3, y_5)$

Table 5. Satisfaction of C2 by All Groups Spanning T_3

Table 5 lists the variables in the spanning exclusive groups of T_3 and the partial of J_3 with respect to the exclusive group freedoms. Due to the positivity of design variables we can calculate the ranges of partial derivatives in the given domain and we find that T_3 is monotonic with respect to the variables in Y_3 and Y_7 and these variables are always driven to their limits by T_3. Hence, the exclusive group Y_3 and Y_7 decisions satisfy C2.2. We can also show that Y_3 is an order invariant exclusive group. The variables in Y_3 occur only in T_3 and so are driven to the same limit in every order because only T_3 can decide them. (Y_3 satisfies condition C1.2). Y_7 is not order invariant because the variable in Y_7 occurs in all tasks and with different monotonicities; in different orders it can be driven to different values by different tasks. We can also see that the exclusive group Y_5 decision is sensitive only to the order invariant exclusive group Y_3 (apart from itself) and thus Y_5 is also forming task T_3 invariant (C2.1). All exclusive groups spanning T_3 are forming task T_3 invariant and thus there is no need to re-execute T_3 because, its decisions under any order will be the same as its independent decisions. Such tasks, all whose spanning exclusive groups are forming task invariant, will be referred to as *tasks with invariant spanning groups*.

Notice that identifying invariant exclusive groups using C1 or C2 is simple because it just requires mapping design variables into exclusive groups and checking the sign invariance of the range of first partial derivatives (using tools such as interval analysis) or evaluating the mixed partials with respect to the exclusive groups. It is also interesting to observe that if a particular exclusive group is sparse (or the number of forming tasks is minimal) then it is likely to be order invariant, because the number of deciding tasks is less. Such is the case with the order invariant DC motor exclusive group Y_3. If the exclusive group is monotonic in a task, then it is (forming) task invariant. If an exclusive group is both monotonic and sparse then it is a strong candidate for order invariance. Thus the topological notion of sparseness and the analytical sensitivity based idea of monotonicity come together in a synergistic fashion to enhance the two invariances. These ideas are useful in ordering decision making in cross-functional teams in the following incremental fashion:

- If a certain exclusive group is order invariant, then the size (and thereby the complexity) of all tasks are reduced. If it is forming task invariant, then the size of the particular forming task is reduced.

- All tasks with invariant spanning groups need not be required to make decisions after having been asked to make independent decisions.

- If every task is spanned by invariant groups, as in the case of DC motor, then the quality loss calculation and optimal order identification can be simplified as shown below.

In the next section, we delve in on the invariance of spanning exclusive groups in more detail.

3.5 Composing Partial Quality Losses

The next step in our reasoning involves attributing the quality loss incurred by a task to losses of specific exclusive group freedoms. We will introduce an intermediate construct called Partial Quality Loss. *Partial Quality Loss* (PQL) incurred by a task due to loss of a particular exclusive group freedom is the degradation in the task results due to the loss of only that particular exclusive group freedom. For instance, the partial quality loss incurred by T_3 due to loss of exclusive group Y_7 freedom is the loss in quality of the output of T_3 when design freedom in only the variables belonging to Y_7 is lost. We show in appendix A4 that, if all exclusive groups spanning a task satisfy C1 or C2.1, as is the case with T_3, then the total quality loss incurred by the task is the sum of partial quality losses due to loss of each individual exclusive group freedom. This enables the decomposition of the quality loss of an order into partial quality losses over exclusive groups. However, the value of this partial quality loss depends on which task to which T_3 lost its exclusive group Y_7 freedom. Y_7 freedom could be lost to either T_1 or T_2. This implies that we have to calculate both terms, partial quality loss for loss of Y_7 freedom to T_1 and to T_2, and use the quantity that is appropriate to the order. In other words partial quality loss *incurred* is a function of three entities: 1) the deciding task 2) the freedom losing task and 3) the exclusive group under consideration.

3.5.1 Quality Loss Infliction and Stage Independence

Consider the case where instead of T_3 losing its exclusive group Y_7 freedom to other tasks, other tasks lost the exclusive group Y_7 freedom to T_3. In such cases T_3 "inflicts" a quality loss on other tasks by constraining them with its decision of Y_7. The partial quality loss inflicted by a task such as T_3 is defined to be the weighted sum of the partial quality losses incurred by all other tasks due to loss of exclusive group freedoms to T_3. (When all exclusive groups spanning a task satisfy C1 or C2.1, the partial quality loss incurred due to loss of a particular exclusive group freedom can be readily calculated with just the independent decisions, as shown in Appendix A4.) Because, the tasks forming an exclusive group are known from the topological structure, *the partial quality loss inflicted is only a function of the deciding task and the exclusive group!* How can we use this to our advantage? In designs where every task is spanned by exclusive groups satisfying C1 or C2.1, the quality loss of an order, which equals the sum of partial losses incurred, is shown in Appendix A4 to be equal to the sum of the quality losses inflicted by each task in the order. We can represent these quality losses as a network, as illustrated in Figure 3 for the DC motor.

The nodes of the network (in bold letters) represent the tasks remaining to be executed at a particular stage of the cross-functional decision process. For example the start node 123 indicates that tasks T_1, T_2 and T_3 are left to be executed and the node 23 indicates that tasks T_2 and T_3 are left to be executed. End node signifies that no more tasks are left to be executed. The edge which connects node 123 and node 23 corresponds to the execution of task T_1 in the first stage and the weight of this arc, $QLI_1^{2,3}$, is the quality loss inflicted by T_1 on T_2 and T_3 when it is the first task to make decisions. Notice that one task is executed at every stage and each directed path from the start node to the end node corresponds to one specific order and the length of the path equals the sum of quality losses inflicted by each of the tasks. All orders are represented in the network, the number of paths from start to end equals the number of orders. If every task is spanned by exclusive groups satisfying C1 or C2.1, the quality loss of the order is equal to the sum of quality losses inflicted, which is equal to the length of the path from the start to the end node. Hence the optimal order (with the smallest quality loss) corresponds to the shortest path of the network given in Figure 3.

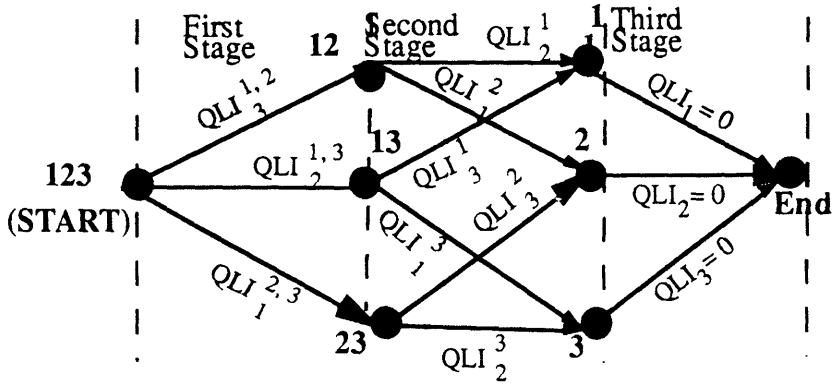

Figure 3: Network Representing the Cross-Functional Decision Orders

4. Identifying the Optimal Order

The procedure we use to identify the optimal order involves five steps, as described below:

Step 1 We execute each design task separately to get its "independent" output and decisions.

Step 2 We partition the design variables into $M = 2^n - 1$ exclusive groups, $y_1, y_2, \cdots, y_M$.

Step 3 We verify if a specific exclusive group is forming task invariant or order invariant. Any task all whose spanning exclusive groups are invariant need not be executed again, after having executed it once in Step 1. Its decisions at a stage of the order are the same as its independent decisions. Also the quality loss of such a task is a simple algebraic summation of the partial quality losses.

Step 4 When every task is spanned by exclusive groups satisfying C1 or C2.1, we store in a matrix (Table 6), the quality losses inflicted by each task (by deciding the variables in each spanning exclusive group). We call the matrix, Quality Loss Matrix(QLM). QLM(i,j) denotes the quality loss inflicted by T_j by deciding the variables in Y_i. Notice that a task cannot decide the values of design variables in exclusive groups which it does not form. For example T_1 cannot decide the value of variables in Y_2 and therefore cannot inflict a quality loss. QLM(i,j) is set to x if T_j cannot inflict a quality loss in Y_i.

Notice that if a particular exclusive group is order invariant, then the corresponding row in the QLM will be zero (except for the x's denoting the non-forming tasks). This is because the exclusive group does not contribute to any quality loss. Such an exclusive group may be removed from the design problem, reducing the size and complexity. In Table 6, this is the case with exclusive groups Y_1, Y_2, Y_3, Y_4 *and* Y_6. (It is noteworthy that Y_1, Y_2 and Y_3, are also the sparse exclusive groups, formed by a single design task).

	T_1	T_2	T_3
Y_1	0	x	x
Y_2	x	0	x
Y_3	x	x	0
Y_4	0	0	x
Y_5	x	0.602	0.188
Y_6	0	x	0
Y_7	0.001	0.001	1.081

Table 6: Quality Loss (Inflicted) Matrix

Step 5 Our next step involves determination of the optimal order using the quality loss terms stored in QLM. We construct a network of quality losses, similar to Figure 3, and using the terms in QLM(see Figure 4). The arc connecting node 123 and node 23 corresponds to the execution of task T_1 in the first stage and the weight of this arc is the quality loss inflicted by T_1 when it is the first task to make decisions (= QLM(7,1) + QLM(4,1) + QLM(1,1)). Similarly, the arc connecting nodes 23 and node 3 corresponds to the execution of task T_2 in the second stage and its weight is the quality loss inflicted by T_2 when it is the second task to make decisions (= QLM(5, 2)). The weights of the other arcs can be filled in the same fashion. It is noteworthy that weights of several arcs (italicized) equal zero due to order invariant exclusive groups. Since the optimal order is the

one with the lowest sum of the quality loss inflicted at all stages it is obtained by finding *the shortest path from the start node to the end node* (using an algorithm like dynamic programming [1]).

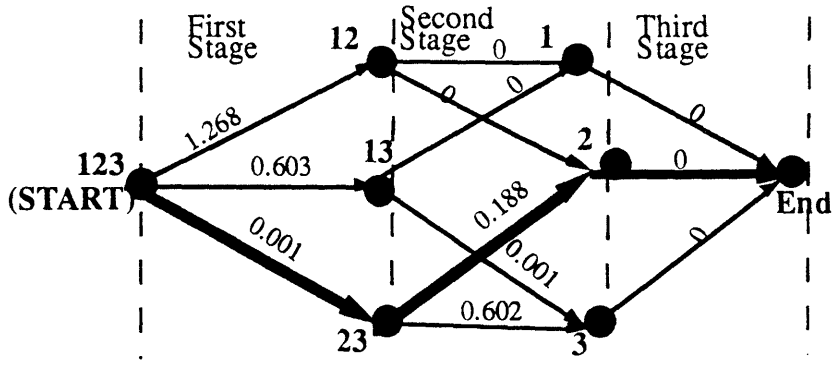

Figure 4: Network with Quality Losses Inflicted Computed from QLM (Bold lines show the shortest path which is also the optimal order.)

5. Conclusion: Strengths and Limitations of the Approach

Our analysis of decision ordering in cross-functional teams has served several purposes which are reviewed here.

Partitioning of design variables into exclusive groups has enabled the decomposition of quality loss incurred by a task into components due to loss of exclusive groups. In real design decision making processes, where the number of tasks or lifecycle considerations, n is much less than the number of design variables, m, such a decomposition would result in substantial computational payoffs. It is noteworthy that we are interested in seeking special solutions to a nonlinear program, solutions that correspond to ordered decision making in a cross-functional group. Clearly, such solutions (to a more constrained problem) will be less optimal than the globally optimal solution to the nonlinear program.

In a three step approach exclusive group structure has been exploited repeatedly to arrive at a series of sufficient conditions. First, conditions are identified, under which the exclusive group is order invariant and forming task invariant. Such groups reduce the complexity of one or more design tasks. When all exclusive groups spanning a task satisfy C1 or C2.1, quality loss incurred by the task can be written as the summation of quality loss components due to loss of individual exclusive groups. In the third step, quality loss is interpreted as loss inflicted to induce stage independence in calculating the optimal order so that the shortest path of a directed network, equals the optimal decision order.

This analysis has interwoven connectivity and sensitivity issues together in identifying the optimal order. Partitioning into exclusive groups is done on the basis of topological connectivity. However, the concept of invariance (order invariance and forming task invariance) uses both connectivity and sensitivity. As observed earlier, both sparseness (topological notion) and monotonicity (sensitivity based idea) enhance invariance. The reasoning process reveals the utility of both these aspects and unearths the strong links between design structure and optimal decision strategies.

The results are incremental in their utility. If a certain exclusive group is forming task invariant, then the size of the particular forming task is reduced. If it is order invariant, then the size (and thereby the complexity) of all tasks are reduced. If all exclusive groups spanning a task are invariant, then the task is not required to make decisions after having been made independent decisions. If all exclusive groups spanning every task are invariant, as in the case of DC motor, then the optimal order is obtained simply by identifying the shortest path in a network of quality losses.

Although the conditions stated to identify invariance apply only to specialized situations, the notion of quality loss, exclusive groups and invariance are broader and can be applied in more general cases, in particular, to our next step study of iterative decision strategies.

Also, there are several limitations to the approach. The results hold for design processes with specially structured design tasks. Tasks are nonlinear programs with a continuously differentiable objective. Constraints could only be either range inequalities or equalities that could be explicitly eliminated by symbolic propagation (as in serial constraint networks). The possibility of including other type of constraints in the objective, other definitions for Quality Loss and methods for weight determination have to be investigated in future.

Our definition of a task assumes that a task is able to decide design variables by itself, without assistance from other tasks. This model may be invalid in some design situations. Finally, the conditions developed to recognize invariances are not strong in non-sparse, non-monotonic situations.

Acknowledgments

We gratefully acknowledge the funding for this research from the National Science Foundation and from the MIT Leaders for Manufacturing Program. We have received much helpful critique from several colleagues, including Tom Magnanti, Dimitris Bertsimas, and Viên Nguyen.

Bibliography

[1] R. Bellman and S. E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, 1962.

[2] K. B. Clark and T. Fujimoto. *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*. Harvard Business School Press, Boston, 1991.

[3] E. Colburn. *Abstracting Design Relations*. Masters Thesis, Mechanical Engineering Department, Carnegie Mellon University, 1988.

[4] S. D. Eppinger, D. E. Whitney, R. P. Smith and D. A. Gebala. *Organizing the Tasks in Complex Design Projects*. ASME Design Theory and Methodology Conference, Chicago, pages 39-45, September 1990.

[5] D. Gebala. *Improving Design Activities through Model-based Analysis*, Master's Thesis, Massachusetts Institute of Technology, 1991.

[6] D. G. Pekar. *Modeling Design Time and Quality Tradeoffs in Automotive Component Design*, Master's Thesis, Masachusetts Institute of Technology, 1992.

[7] S. D. Roodvoets. *An Evaluation of the Influence of Platform Team Organization On Product Development Performance*, Master's Thesis, Massachusetts Institute of Technology, 1991.

[8] H. A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 2nd edition, 1981.

[9] R. P. Smith and S. D. Eppinger. *Modeling Design Iteration*. MIT Sloan School of Management Technical Report 3160-90, June 1990.

[10] J. Sobieszczanski-Sobieski. *Multidisciplinary Optimization for Engineering Systems: Achievements and Potential*, NASA Technical Memorandum 101566, 1989.

Appendix A1. Mathematical Description of a Cross-Functional Design Task

In our model, we assume the following:

• Each task T_i is a nonlinear program with a continuously differentiable objective J_i. J_i is explicitly expressible as a function of the design variables. Let $\mathbf{Z_i}$ be a set consisting of the design variables appearing in J_i and z_i be a vector consisting of these variables.

• Finite upper and lower limits are specified on each design variable so that the domain of design variables is a closed and bounded (compact) set.

• All equality constraints have been explicitly eliminated or used for symbolic propagation. The only inequalities given are lower and upper limit constraints.

• Suppose that variables $z_{i1}, z_{i2}, \cdots, z_{ir}$ occur in task T_i. Without loss of generality consider the order $\varphi = \{T_1, T_2, T_3, \ldots, T_n\}$. The nonlinear programming problem for task T_i (in the order φ) and the corresponding optimality conditions are stated in Table A1.

Mathematical representation of T_i:	K-T Optimality conditions for task T_i in the order φ.
T_i: Min $J_i(z_i)$ subject to $L_{ik}(z_{ik}) = l_{ik} - z_{ik} \leq 0$ $U_{ik}(z_{ik}) = -u_{ik} + z_{ik} \leq 0$ $S_{iq}^{\varphi} = z_{iq} - z_{iq}^{\varphi} = 0$ for $k = 1, 2, ..., r$ and $z_{iq} \in \mathbf{Z}_i \cap \{\mathbf{Z}_1 \cup \cdots \cup \mathbf{Z}_{i-1}\}$ $q = 1, 2, ..., Q$ • l_{ik} and u_{ik} are the limits on z_{ik} • z_{iq} represents the design freedom lost by T_i in the order φ. z_{iq}^{φ} is the value set by a preceding task.	$\nabla J_i(z_i^*) + \sum_{k=1}^{r} \lambda_{ik} \nabla L_{ik}(z_i^*) + \nu_{ik} \nabla U_{ik}(z_i^*) + \sum_{l}^{Q} \kappa_{iq} \nabla S_{iq}^{\varphi} = 0$ $\lambda_{ik} L_{ik}(z_i^*) = 0;$ $\nu_{ik} U_{ik}(z_i^*) = 0;$ $\lambda_{ik}, \nu_{ik} \geq 0; \; k = 1, 2, ..., r$ where $\nabla J_i = \begin{Bmatrix} \frac{\partial J_i}{\partial z_{i1}} \\ \vdots \\ \frac{\partial J_i}{\partial z_{ir}} \end{Bmatrix}$ and λ, ν and κ are the lagrange multipliers.

Table A1. Kuhn-Tucker Optimality Conditions

We make the following observations. (' superscript indicates the vector transpose).

• All the components of the gradients of L_{ik} and U_{ik} are zero, but for kth component:

$\nabla L_{ik} = \{0, 0, \ldots, 0, -1, 0, 0, \ldots, 0\}'$; $\nabla U_{ik} = \{0, 0, \ldots, 0, 1, 0, 0, \ldots, 0\}'$

Also $L_{ik}(z_i^*) = L_{ik}(z_{ik}^*)$; $U_{ik}(z_i^*) = U_{ik}(z_{ik}^*)$;

• If in an order T_i decides the value of design variable z_{is}, then the order constraints, S_{iq}^{φ}, do not affect z_{is}.

$$z_{is} \notin \mathbf{Z}_i \cap \{\mathbf{Z}_1 \cup \cdots \cup \mathbf{Z}_{i-1}\}. \text{ So } \frac{\partial S_{iq}^{\varphi}}{\partial z_{is}} = 0$$

For variables such as z_{is}, decided by T_i, we can rewrite the K-T conditions as follows:

$\nabla J_i(z_i^*) - \lambda_{is} + \nu_{is} = 0$

$\lambda_{is} L_{is}(z_{is}^*) = 0;$

$\nu_{is}\ U_{is}(z_{is}^*) = 0;$

$\lambda_{is},\ \nu_{is} \geq 0;$

It is noteworthy that the K-T conditions for z_{is} are still a function of z_i^*.

• We will combine the K-T conditions for all variables belonging to the same exclusive group $\mathbf{Y}_j$, decided by T_i. Suppose that exclusive group $\mathbf{Y}_j$ consists of variables $z_{i1}, z_{i2}, \cdots, z_{is}$.

Let $y_j = \begin{Bmatrix} z_{i1} \\ \vdots \\ z_{is} \end{Bmatrix}$; $\pi_{ij} = \begin{Bmatrix} \lambda_{i1} \\ \vdots \\ \lambda_{is} \end{Bmatrix}$; $\rho_{ij} = \begin{Bmatrix} \nu_{i1} \\ \vdots \\ \nu_{is} \end{Bmatrix}$; $\frac{\partial J_i}{\partial \mathbf{y}_j} = \begin{Bmatrix} \frac{\partial J_i}{\partial z_{i1}} \\ \vdots \\ \frac{\partial J_i}{\partial z_{is}} \end{Bmatrix}$ be column vectors.

Now we can write (K-T) conditions for the exclusive group $\mathbf{y}_j$ as:

$\frac{\partial J_i}{\partial \mathbf{y}_j}(z_i^*) - \pi_{ij} + \rho_{ij} = 0$

$\lambda_{ik} L_{ik}(z_{ik}^*) = 0;$

$\nu_{ik}\ U_{ik}(z_{ik}^*) = 0;\ \lambda_{ik},\ \nu_{ik} \geq 0;\ k = 1, 2, \ldots, s$

Appendix A2. Order Invariant Exclusive Groups

An exclusive group Y_j is said to be order invariant if the values of each design variable belonging to Y_j is the same under any order.

Proposition 1: An exclusive group Y_j is order invariant if C1.1 or C1.2 is satisfied.

C1.1 $$\frac{\partial}{\partial y_k}\frac{\partial J_p}{\partial y_j} = 0 \ \forall \ y_k, (j \neq k) \text{ spanning every } T_p \in t_j.$$

(i. e. Y_j is insensitive to all other spanning exclusive groups in every forming task)

and the independent decisions of all forming tasks equal the same value y_j^*.

C1.2 $\forall \ X_i \in Y_k$, either $\frac{\partial J_p}{\partial X_i} > 0 \ \forall T_p \in t_j$ or $\frac{\partial J_p}{\partial X_i} < 0 \ \forall T_p \in t_j$

Proof:

Consider a order in which the value of the design variables in y_j is decided by one of the tasks $T_p \in t_j$. All orders have this property because by definition of decision in Section 3, the value of the design variables in y_j can only be decided by one of the tasks $T_p \in t_j$. Now if C1.2 is satisfied, due to monotonicity, any design variable will be driven to the same decision value y_j^* in every order (this value equals one of its limits based on its monotonicity).

If all $\frac{\partial J_p}{\partial X_i} > 0$, then $\frac{\partial J_i}{\partial y_j} > 0$. To satisfy K-T conditions, $\pi_{ij} > 0$ and $L_{ik}(z_{ik}^*) = 0$. So in all orders, $z_{ik} = l_{ik}$. If all $\frac{\partial J_p}{\partial X_i} < 0$, then $\frac{\partial J_i}{\partial y_j} < 0$. Now $\rho_{ij} > 0$ and so, $U_{ik}(z_{ik}^*) = 0$, $z_{ik} = u_{ik}$.

If C1.1 is satisfied, then we first prove that the value decided by any task $T_p \in t_j$ for the design variables in y_j is always the same as the task's independent decision.

$$\text{When } \frac{\partial}{\partial y_k}\frac{\partial J_p}{\partial y_j} = 0 \ \forall \ y_k, (j \neq k) \text{ spanning } T_p \in t_j \quad \frac{\partial J_i}{\partial y_j}(z_i^*) = \frac{\partial J_i}{\partial y_j}(y_j^*)$$

Now we can write (K-T) conditions for the exclusive group y_j as:

$$\frac{\partial J_i}{\partial y_j}(y_j^*) - \pi_{ij} + \rho_{ij} = 0$$

$\lambda_{ik} L_{ik}(z^*_{ik}) = 0;$

$\nu_{ik}\ U_{ik}(z^*_{ik}) = 0;\quad \lambda_{ik},\ \nu_{ik} \geq 0;\quad k = 1, 2,..., s$

Now one observes that the K-T conditions are independent of other exclusive groups and independent of order! Thus the K-T conditions "separate out" for any exclusive group. So the (value decided by a task when it makes its) independent decision satisfies the K-T conditions under any order. This is because when the task makes its independent decision it has to satisfy the same K-T conditions that it does under any order. Now, if the independent decisions of every forming task are the same, as the proposition states, then y_j is order invariant.

Appendix A3. Task Invariant Exclusive Groups

Proposition 2: Exclusive group Y_j is forming task T_i invariant if it satisfies C2.1 or C2.2

C2.1 $\frac{\partial}{\partial y_k}\frac{\partial J_i}{\partial y_j} = 0 \ \forall \ y_k, (j \neq k)$ spanning T_i and not satisfying C1.

(i. e. Y_j decision by T_i is sensitive only to other provenly order invariant exclusive groups).

C2.2 $\forall \ X_i \in Y_k$, either $\frac{\partial J_p}{\partial X_i} > 0$ or $\frac{\partial J_p}{\partial X_i} < 0$ (i. e. every Variable in y_j is monotonic in T_i in the range of the given design problem).

Proof: It is easy to show that if T_i makes Y_j decision and y_j is monotonic in T_i (C2.2) then the variables in Y_j will be driven to the same value (one of the limits; similar to the argument used in C1.2). This can be verified using K-T conditions for y_j, since there are no order constraints affecting the design variables in y_j.

If C2.1 is satisfied, then again the K-T conditions, written for Y_j become order independent, in a fashion similar to the Appendix A2 (we set all order invariant exclusive groups to their constant, order invariant decision value). Hence the value decided by T_i for the design variables belonging to Y_j is the same in every order in which T_i makesY_j decision. This order invariant value equals the independent decision of T_i because the independent decision is the same as the decision of the order in which T_i makes decisions first.

Appendix A4. Partial Quality Loss

A task T_i is said to lose the exclusive group Y_k freedom in a order φ if the variables inY_k are decided by a preceding task in the order φ.

Partial quality loss incurred by T_i due to loss of exclusive group Y_k freedom in a order φ, QL_{ik}^{φ}, is defined as the quality loss incurred by T_i when it loses only the exclusive group Y_k freedom. Mathematically:

$$QL_{ik}^{\varphi} = J_i(y_1^*, y_2^*, \cdots, y_k^{\varphi}, \cdots, y_L^*) - J_i^*$$

where $y_1, y_2, \cdots, y_k, \cdots, y_L$ are assumed to be the vectors with the variables in the corresponding exclusive groups spanning T_i.

Proposition 3:

If all exclusive groups spanning a task T_i satisfy either C2.1, C1.1 or C1.2, then Quality Loss incurred by task T_i in a order φ, QL_i^{φ}, can be written as the sum of partial quality losses incurred for loss of each exclusive group freedom in order φ.

Proof: Quality loss in φ can be written as: $QL_i^{\varphi} = J_i(y_1^*, y_2^{\varphi}, \cdots, y_k^{\varphi}, \cdots, y_L^*) - J_i^*$

(assuming the degrees of freedom lost in φ are in $y_2, \cdots, y_k$)

Applying Taylor's Theorem for multiple variables (given below) to the quality loss term:

$$f(x + \Delta x, y + \Delta y) - f(x, y) = \left[\Delta x \frac{\partial f}{\partial x} + \Delta y \frac{\partial f}{\partial y}\right] + \frac{1}{2!}\left[(\Delta x)^2 \frac{\partial^2 f}{\partial x^2} + \Delta x\, \Delta y \frac{\partial^2 f}{\partial x \partial y} (\Delta y)^2 \frac{\partial^2 f}{\partial y^2}\right] + \cdots$$

$$QL_i^{\varphi} = J_i(y_1^*, y_2^{\varphi}, \cdots, y_k^{\varphi}, \cdots, y_L^*) - J_i(y_1^*, y_2^*, \cdots, y_k^*, \cdots, y_L^*)$$
$$= \left[(y_2^{\varphi} - y_2^*)\frac{\partial J_i}{\partial y_2} + \cdots + (y_k^{\varphi} - y_k^*)\frac{\partial J_i}{\partial y_k}\right] +$$
$$\frac{1}{2!}\left[(y_2^{\varphi} - y_2^*)^2 \frac{\partial^2 J_i}{\partial y_2^2} + \cdots + (y_k^{\varphi} - y_k^*)^2 \frac{\partial^2 J_i}{\partial y_k^2} + \text{cross-derivative terms}\right] + \cdots$$

When C1.1 or C1.2 is satisfied then the exclusive groups are order invariant in which case the Δx term is zero. If C2.1 is satisfied, then the cross-derivative terms are zero with respect to all groups that do not satisfy C1.1 or C1.2. Applying these in the quality loss term, we have:

$$QL_i^{\varphi} = QL_{i2}^{\varphi} + \cdots + QL_{ik}^{\varphi}, \text{ where } QL_{ik}^{\varphi} = (y_k^{\varphi} - y_k^*)\frac{\partial J_i}{\partial y_k} + \frac{1}{2!}(y_k^{\varphi} - y_k^*)^2 \frac{\partial^2 J_i}{\partial y_k^2} + \cdots$$

Thus the quality Loss incurred by task T_i in a order φ, QL_i^{φ}, is the sum of partial quality losses incurred for loss of each exclusive group freedom in order φ.

Corollary If the exclusive groups spanning every task in a product development process satisfies the conditions of Proposition 3, then the quality loss of a sequence is the weighted sum of partial quality loss of each individual tasks over the exclusive group freedom lost in the sequence. (Over the exclusive group freedoms not lost, the quality loss is zero, so adding such exclusive groups does not affect the summation).

From Proposition 3 and quality loss definition, $QL^{\varphi} = \sum_{i=1}^{n} w_i \sum_{k=1}^{M} QL_{ik}^{\varphi}$

Reversing the order of summation, $QL^{\varphi} = \sum_{i=1}^{n} w_i \sum_{k=1}^{M} QL_{ik}^{\varphi} = \sum_{k=1}^{M} \sum_{i=1}^{n} w_i \; QL_{ik}^{\varphi}$

Defn: Quality loss inflicted by a task T_h by deciding the exclusive group y_j in a sequence φ, QLI_{hj}^{φ}, is defined as the weighted sum of the partial quality losses incurred by each task which loses exclusive group y_j freedom to T_h in the sequence φ (tasks which do not lose the exclusive group freedom incur a loss of zero; hence the summation can include all tasks): $QLI_{hj}^{\varphi} = \sum_{i=1}^{n} w_i \; QL_{ij}^{\varphi}$

Defn: Quality loss inflicted by a task T_h, QLI_h^{φ}, is defined as the sum of the partial quality losses inflicted due to all the exclusive groups decided by T_h in the sequence φ.
(By definition of ordered decision making, each exclusive group decision is made by one task, which inflicts quality losses on all other tasks. So the sum of quality losses inflicted due to all the exclusive groups is the same as the sum of quality losses inflicted by all the tasks. In other words, $\sum_{k=1}^{M} QLI_{hk}^{\varphi} = \sum_{h=1}^{n} QLI_h^{\varphi}$

From corollary above we have, $QL^{\varphi} = \sum_{k=1}^{M} \sum_{i=1}^{n} w_i \; QL_{ik}^{\varphi}$

Using $QLI_{hj}^{\varphi} = \sum_{i=1}^{n} w_i \; QL_{ij}^{\varphi}$, we have, $QL^{\varphi} = \sum_{k=1}^{M} QLI_{hk}^{\varphi} = \sum_{h=1}^{n} QLI_h^{\varphi}$

This shows that the quality loss of a sequence is also the sum of the quality losses inflicted by each task in the sequence.