

# Integration Schemes and Decomposition Modes in Complex Problem Solving

**Author:**

Efatmaneshnik, Mahmoud; Reidsema, Carl

**Event details:**

System Engineering, Test and Evaluation (SETE2008)  
Canberra, Australia

**Publication Date:**

2008

**DOI:**

<https://doi.org/10.26190/unsworks/390>

**License:**

<https://creativecommons.org/licenses/by-nc-nd/3.0/au/>

Link to license to see what you are allowed to do with this resource.

Downloaded from <http://hdl.handle.net/1959.4/38291> in <https://unsworks.unsw.edu.au> on 2024-04-20

# **Integration Schemes and Decomposition Modes in Complex Problem Solving**

**Mahmoud Efatmaneshnik, Carl Reidsema**

Design Research Laboratory

School of Mechanical and Manufacturing Engineering

The University of New South Wales

Sydney NSW 2052 Australia

mahmoud@student.unsw.edu.au , reidsema@unsw.edu.au

## **ABSTRACT**

Integrated product development requires that decomposition and integration schemes be congruent and in harmony with each other. This harmony can mark complex and large scale product development project with success. In domain of problem solving one needs to resort to a priori knowledge about problem structure, and its underlying couplings/complexity. Here, we refer to the problem structure as the self map of the system. Usually for large scale problems the self needs to be decomposed for tractability purposes the structure of the problem after decomposition is the real structure to be dealt with. In this paper we measure the complexity of systems before and after decomposition. We refer to complexity of the system/problem before decomposition as self complexity and complexity of system after decomposition as real complexity. It is reasoned that real complexity cannot be less than the self complexity. It is also noted that laws of weak and strong emergence can be demonstrated by using real complexity measure of decomposed system. This would have important implications in problem classification and choosing the right design process that is in congruence with complexity of the problem.

## **Introduction**

A product or system at the development level can be modelled by the set of design variables. The design variable sets include subsets of sizing variables, shape variables, topologies and process knowledge and manufacturing variables such as process capabilities (Prasad, 1996). Formal definition of design problem solving constitutes the process of assigning values to variables in accordance with the given design requirements, constraints, and optimization criterion (Zdrahal and Motta, 1996). For example in design of an aircraft, assigning values of sizing to wing, body and control surfaces is regarded as a design activity. The relationship between design variables is summarized in Parameter based Design Structure Matrix (PDSM). PDSM is used for modeling low-level relationships between design decisions and parameters, systems of equations, subroutine parameter exchanges which represents the product architecture (Browning, 2001). For example PDSM of the aircraft at the conceptual level might address how the wing configuration as a design variable affects cruise speed, fuel consumption, sizing variables, or other design variables of the aircraft. PDSM is upstream information in design process about the product and is achieved through design of experiments, expert suggestions, design of experiments and simulation techniques in

particular statistical Monte Carlo Simulation, and combination of some/all of these techniques (Table 1). PDSM can be represented via a graph to which we refer as the self map of the product /system (figure 1).

Table 1. A PDSM is a weighed adjacency matrix. This PDSM has 10 Variables.

-	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	0.76	0.45	0.16	0.22	0.77	0.12	0.01	0	0
V2	0.76	0	0.11	0.65	0.44	0.78	0	0	0	0.18
V3	0.45	0.11	0	0.64	0.11	0.31	0.02	0	0.15	0
V4	0.16	0.65	0.64	0	0.45	0.34	0	0	0	0
V5	0.22	0.44	0.11	0.45	0	0	0	0.01	0	0.01
V6	0.77	0.78	0.31	0.34	0	0	0	0	0	0
V7	0.12	0	0.02	0	0	0	0	0.2	0.7	0.1
V8	0.01	0	0	0	0.01	0	0.2	0	0.2	0.8
V9	0	0	0.15	0	0	0	0.7	0.2	0	0.9
V10	0	0.18	0	0	0.01	0	0.1	0.8	0.9	0

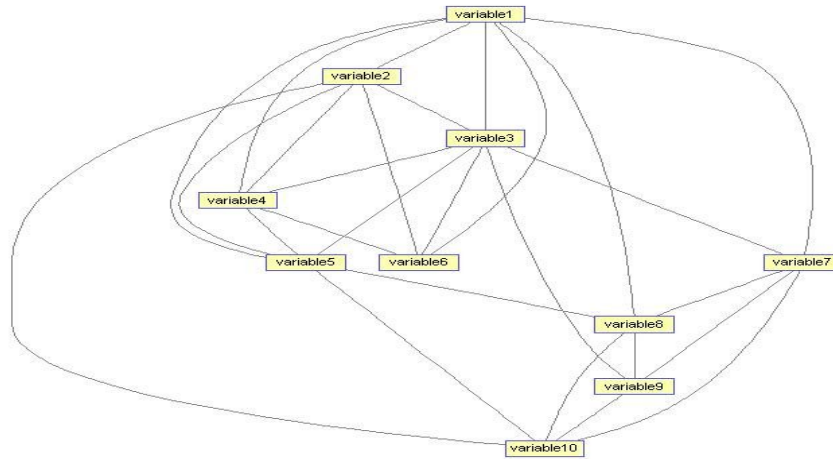


Figure 1. The self map of problem/system.

Structural decomposition of a problem takes decomposition as the clustering of the different design variable or parameters, objectives and constraints into subsystems (Browning, 1997). Decomposition of a product is synonymous to decomposition of system/product's PDSM. Table 2 and figure 2 show one possible decomposition of the example system in figure 1. The design variables could have been grouped in many different ways. Ulrich and Eppinger (2004) define the architecture of a product as the scheme by which the decomposed elements are arranged in chunks. The choice of product architecture has broad implications for product performance, product change, product variety, and manufacturability (Ulrich and Eppinger, 2004). A desirable decomposition is one that leads to least amount of interactions in between the subsystems. In literature this is referred to as optimal decomposition (Michelena Papalambros, 1997) and robust decomposition (Browning, 1999). Appropriately clustering interdependent design parameters can reveal a preferred integration of low-level activities into higher-level ones (Browning, 1997).

Table 2. The variables of Table 1 are rearranged to form three subsystems.

		Subsystem1			Subsystem2				Subsystem3		
		V5	V4	V2	V10	V8	V7	V9	V6	V1	V3
Subsystem 1	V5	0	0.45	0.44	0	0	0.02	0.02	0.53	0.22	0.11
	V4	0.34	0	0.65	0	0	0	0	0.43	0.16	0.64
	V2	0.3	0.12	0	0	0	0.2	0.1	0.2	0.76	0.12
Subsystem 2	V10	0.01	0	0.18	0	0.8	0.1	0.9	0	0	0
	V8	0.01	0	0	0.1	0	0.2	0.4	0	0.01	0
	V7	0	0	0	0.3	0.45	0	0.1	0	0.12	0.02
	V9	0	0	0	0.5	0.2	0.7	0	0	0	0.15
Subsystem 3	V6	0	0.34	0.78	0	0	0	0	0	0.77	0.31
	V1	0	0	0.53	0	0	0	0	0.1	0	0.32
	V3	0	0	0.11	0.72	0	0.3	0.52	0.2	0.45	0

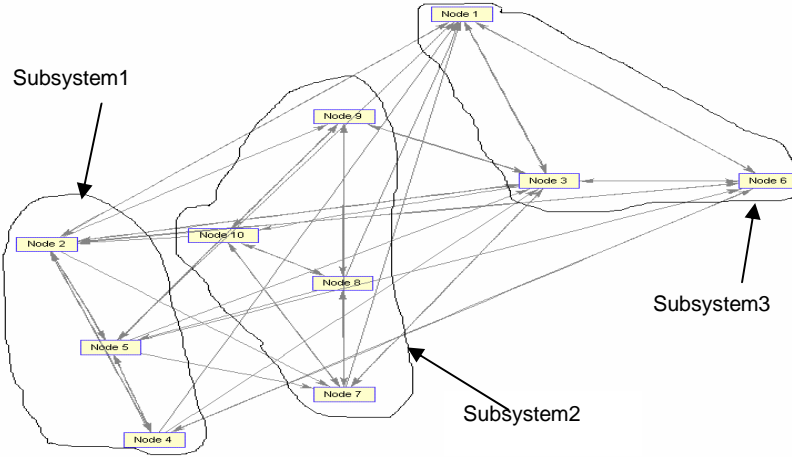


Figure 2. One possible decomposition through clustering the design variables.

The product architecture has a large influence on the appropriate structure of the product development organization since organizational elements are typically assigned to develop various product components (Browning, 1997). Product integration is carried out by gaining a priori information about PDSM and then aligning problem structure and product development organization's structure. To align problem and organizational development two types of planning can be devised:

1. *Bottom up planning for flexible organizational structure* that is to form the design teams subsequent to product decomposition; Tanaka et al. (2000) for example took this approach in the context of distributed problem solving and referred to it as "multi agent system creation". This approach has also been used in a manufacturing system MetaMorph (Maturana et al.,1999) that could dynamically change its form to mimic the task structure. In this case the number of the subsystems may be maximized for better overall performance.

2. *Top-down planning for fixed organizational structures* is to decompose the product in a way that suits the organizational structure; this is used when the organizational

structure is fixed and solid. A greater use of coordination activities between the design teams and/or the use of integration teams is resulted. In this case number of the subsystems is determined according to the number of design teams.

The literature concerning methods of decomposition and integration in design community is not very extensive. Pimmler and Eppinger (1994) explained that for a complex product, such as an automobile, a computer, or an airplane, there are thousands of possible decompositions which may be considered; each of these alternative decompositions defines a different set of integration challenges (Pimmler and Eppinger, 1994) at the organizational level. Pimmler and Eppinger (1994) presented an extensive literature review concerning system decomposition and architecture at the system definition stage of product design; according to them the core research began with Alexander (1964), who described a design process which decomposed (or partitions) designs into minimally coupled groups. Simon (1981) continued by suggesting that complex design problems could be described in terms of hierarchical structures consisting of nearly decomposable systems organized such that the strongest interactions occur within groups and only weaker interactions occur among groups. Henderson and Clark (1990) related the importance of architecture by noting that established firms frequently fail when confronted by a novel architecture. McCord and Eppinger (1993) described a methodology using interactions between components to structure system teams in a development project.

All previous researchers dealt with the methods of decomposition and how the alignment of the subsystems and design groups structure helps the problem solving procedure. This paper takes this discussion to a new level by making three contributions to previous works in the field of decomposition and integration. First we attempt to gather and present all the problem decomposition modes scattered in the literature. Second by attributing a complexity measure to every specific decomposition (real complexity), we measure the integration effort that the particular decomposition pose in front of system integrators. Thirdly it is shown that this measure can be used to determine whether the system has strong or weak emergent properties. The paper ends with a description of several integration schemes aligned with the decomposition modes. It is shown that in case the system has strong emergence, integration based on team collaboration techniques may not be achieved. Instead integration based on holistic solutions gained from various design groups competing and working in parallel on the same problem (Bar Yam, 2004) suit the systems with strong emergence.

## **Decomposition Modes**

According to Papalambros (2002) decomposition of large-scale design problems allows for:

- conceptual simplification of the system
- reduction in the dimensionality of the problem
- more efficient computational procedures
- utilization of different solution techniques for individual sub-problems
- simultaneous design, modularity, multi-objective analysis
- efficient communication and coordination among the diverse groups involved in the design process

Problem decomposition and partitioning of the self map of the system fits in area of graph partitioning. A graph  $G$  is specified by its vertex set,  $V = \{1, \dots, n\}$ , and edge set  $E$ . The total number of nodes in  $G$  is referred to as order of  $G$ . Number of edges in  $G$  is the size of  $G$ .  $G$

$(n, m)$  is a graph of order  $n$  and size  $m$ . The most natural matrix to associate with  $G$  is its adjacency matrix,  $A_G$ , whose entries  $a_{i,j}$  is given by:

$$a_{i,j} = \begin{cases} w_{i,j} & (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

For un-weighted graphs all  $w_{i,j}=1$ , and in undirected graphs for all  $(i,j) \in E$ ,  $a_{i,j} = a_{j,i}$ . A bi-partitioning of graph  $G$  is a division of its vertices into two sets or sub-graphs,  $P_1$  and  $P_2$ . Similarly a  $k$ -partitioning is to divide the vertices of the graph into  $k$  non-empty sets  $P = \{P_1, P_2, \dots, P_k\}$ . A graph can be partitioned in many different ways. In the domain of the problem solving, every node or vertex of a graph represents a variable of the system and every edge of the graph suggests that two parameters of the system are dependent on each other. The strength of the relationship between variable is the corresponding edge weight. An undirected graph as the self map of the system indicates that variables affect one another mutually and equally. The sub-graphs can be regarded as subsystems or sub-problems or agents (Kusiak, 1999). The notion of agency implies that the sub-problems are solved more or less independently from each other. Each design team has autonomy to explore parts of the solution space that is of interest to its own assigned sub-problem (agent).

The system (or problem) is fully decomposable if there is no link in between the subsystems. In this case the corresponding design process is fully concurrent: problems are solved separately and solutions are added together. When dependency in between the subsystems exists, conflict may arise. A conflict is when the solution to one sub-problem is in contrast with the solutions to another sub-problem(s). An important conflict resolution technique is negotiation. Negotiation leads to iteration in the design process. A coordinator may be used to monitor the conflict resolution and negotiation process. Obviously a design process with less number of iterations is more desirable. Thus the criteria for decomposition is that sub-systems must be rendered as independent as possible. The graph partitioning that corresponds to this decomposition criteria is referred to as minimum cut/partitioning.

Problem connectivity is the total number of edges in self map of product/problem divided by the total number of possible edges -- that is number of edges of a complete graph with same number of nodes. The total number of possible edges in a complete undirected graph with  $n$  nodes or vertices is


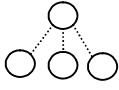
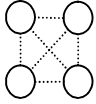
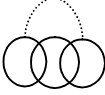
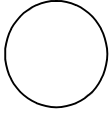
$$K = \binom{n}{2} = \frac{n \times (n-1)}{2} \quad (1)$$

If the self map of PDSM has  $k$  connections (edges), we define the problem connectivity as:

$$p = \frac{k}{K} \quad (2)$$

Surfing the literature related to problem decomposition in the field of distributed problem solving has revealed that depending on the problem's self connectivity decomposition of self map of system/problem can be carried out in several modes (Sosa et al. (2000), Klein et al. (2003), Browning (2001)). These modes are illustrated in table 3. The connectivity values in this table are based the experience of authors with randomly generated graphs. Bearing in mind that it is usually desirable to have subsystems of similar order, the implementation of some of these decomposition modes (in particularly full decomposition mode and integrative mode) may not always be feasible. For problems with denser self map (higher connectivity) modular clustering and overlap decomposition can be used. If the problem's map is very dense and system is really complex then it may not be decomposed at all (Bar Yam, 2004).

Table 3. Decomposition Modes of self map of problems

Connectivity	Very Low (0-0.02)	Low (0.02-0.1)	Intermediate (0.1-0.2)	High (0.2-0.3)	Very High (0.3-1)
Possible or best decomposition strategy	Full decomposition	Integrative Clustering	Modular Clustering	Overlap clustering	No Decomposition
Illustration					

Each of these decomposition modes brings specific strengths, weaknesses and particularity to the problem solving process. As an example an aircraft with blended wing body may not be decomposed completely to separate body and wings with the related design variables being independent or loosely dependent (figure 3). Instead for systems that have subsystems with fuzzy boundaries overlap decomposition may be used; the aligned organizational architecture in charge of conceptual and parametric design of aircraft's body and wings also constitute two different design teams with overlaps and fuzzy boundaries. This is discussed in last section of this paper.

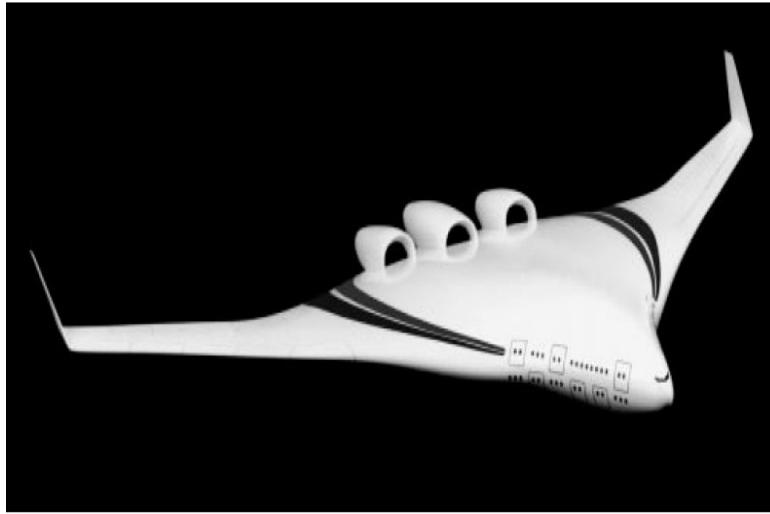


Figure 3. Two components (subsystems) are overlapped in blended wing-body types of aircrafts

In the next section real complexity of decompositions is introduced. Complexity based method clarifies why systems with very high level of connectivity may not be decomposed.

## Real Complexity

In general, complexity is defined as the quality of being intricate and compounded (Duin and Pekalska, 2006). This means that an entity, a problem, a task or a system is complex if it consists of a number of elements (components) related such that it is hard to separate them or to follow their interrelations (Duin and Pekalska, 2006). An entity is more complex if more components and more interdependencies can be distinguished (Duin and Pekalska, 2006). So, Complexity can be characterized by the levels and the kinds of distinction (the variability and the number of elements) and dependency between the components (Duin and Pekalska, 2006). In recent years, there have been many attempts to use complexity measure to facilitate decision-making. For example Suh (2005), defined complexity as a measure of uncertainty in achieving the functional requirements of a design. Yu and Efstathiou (2002) used an entropic complexity measure to indicate the effect of layout modification on manufacturing network. Guenov (2002) presented a complexity measure based on Boltzman's entropy concept to measure the complexity of functional couplings of design parameters system and product functional requirements. Frizelle et al (2002) introduced two types of complexity to be used in supply chain management: flow complexity and stock complexity. Static complexity was introduced as monitoring factor of the system's performance in processing requirements of parts with regards to machine capabilities (Deshmuk et al, 1998). Holttä and Otto (2003) introduced a novel design effort complexity metric that characterized the minimum design effort for the redesign of modules in a modular product design framework.

We used Ontix<sup>1</sup>, the complexity measure embedded in Ontospace software<sup>TM</sup>, that is implicitly described in Marczyk and Dishpande (2006). Ontix is a graph theoretic complexity measure that has the following properties:

1. Complexity of a graph with zero weight nodes and size zero is equal to zero.
2. Complexity is an increasing function of number of edges, their weights, number of vertices, and the weight of vertices.
3. Complexity of a graph with nonzero nodes weights and size zero is non zero and positive.
4. Complexity is an increasing function of number of independent cycles in graph.

Figure 5 demonstrate the above properties of Ontix. Each point in the plots represents a randomly generated graph.

By applying Ontix to PDSM of system, the self complexity of the system is acquired which is the complexity before decomposition. The perceived complexity of the system after it has been decomposed must be different from self complexity. This is so since when a system is simplified it is unavoidable to loose some of the information contained in the system; the amount of information lost results in the increase of an equal amount of relevant uncertainty (Klir, 2003). Any kind of simplification including break of the overall system to subsystems can increase uncertainty (Klir, 2003). More uncertainty about the behavior of a system leads to increased perceived complexity of the system. Thus decomposition increases the overall complexity. There is an easier way to arrive at this conclusion by resorting to the "No Free Lunch Theorem"; it can be stated that facing more overall complexity is the price for having tractable sub-problems.

---

<sup>1</sup> Ontix is the proprietary complexity measure of Ontonix ([www.ontonix.com](http://www.ontonix.com)) a leading complexity management company.



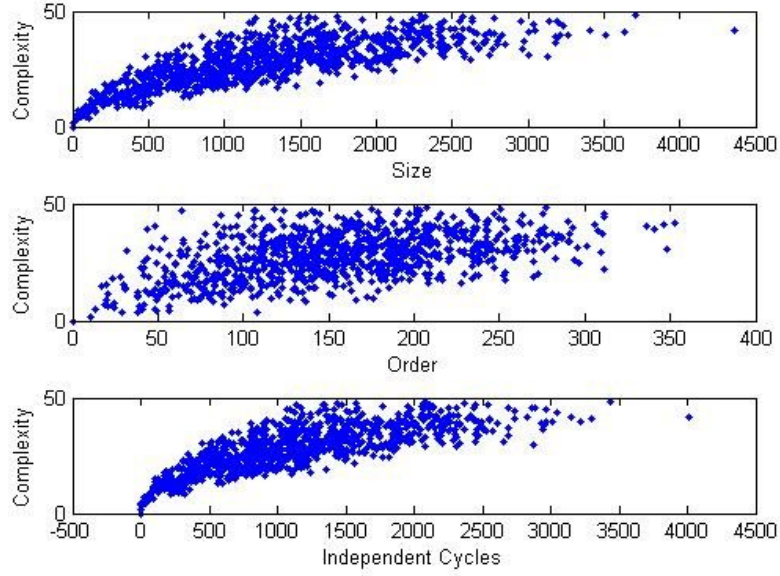


Figure 4. Ontix is an increasing function of graph size, order and number of independent cycles.

We refer to complexity of the systems after it has been decomposed as real complexity of system. A block diagram is the graphical representation of the decomposed system (figure 5). Let  $S$  be the graphical representation of a problem with the adjacency matrix  $A = [a_{i,j}]$  and its complexity  $C(S)$  (self complexity). Consider partitioning  $P$  of the graph  $S$ , into  $k$  sub-graph  $P = \{P_1, P_2, \dots, P_k\}$ . Each of these sub-graphs is a block of the system. Let  $C(P_i)$  be the complexity of each sub-graph determined by the complexity measure. The purpose of decomposition is to reduce the initial problem complexity  $C(S)$  to a number of sub-problems with complexity  $C(P_i)$  less than self complexity  $C(S)$ . Let's define the  $k$  dimensional square matrix  $B$  as the Complexity Based Adjacency Matrix of the Block Diagram with the diagonal entries as the complexity of the sub-graphs (or blocks), and the off-diagonal entries as the sum of the weight of the edges that have one end in each of the two corresponding sub-graphs. The real complexity (induced by decomposition) is the complexity of the block diagram  $C(B)$  and is achieved by applying the graph theoretic complexity measure (Ontix) to matrix  $B$ .

$$B = \begin{bmatrix} C_1 & L_{1,2} & \cdot & \dots & \cdot & L_{1,k} \\ L_{2,1} & C_2 & \cdot & \dots & \cdot & L_{2,k} \\ \cdot & \cdot & \cdot & & & \cdot \\ \vdots & \vdots & & \ddots & & \vdots \\ \cdot & \cdot & & & \cdot & \cdot \\ L_{k,1} & L_{k,2} & \cdot & \dots & \cdot & C_k \end{bmatrix} \quad (3)$$

Where

$$\text{for } \forall i, j = \{1, \dots, k\}: C_i = C(P_i) \text{ and } L_{i,j} = \sum_{i' \in P_i, j' \in P_j} a_{i', j'} \quad (4)$$

Real complexity  $C(B)$  is a subjective measure of the system and is relative to how one might decompose the system. Conversely the self complexity  $C(S)$  is an objective measure of the system and is absolute in being independent from the type of decomposition  $P$ . Real complexity represents the overall complexity of the whole system of subsystems. By expressing the coupled-ness of the system of sub-systems, real complexity represents the integration effort for the whole system after decomposition. The product/system integration efficiency and risk (of not achieving design criteria) is dependent on real complexity as much as it depends on self complexity. Obviously by minimizing the real complexity the integration effort and risk is minimized. We would like to refer to decomposition with minimum real complexity as immune decomposition. Heuristics are required to extract the immune decomposition. A search algorithm based on spectral graph partitioning is presented in Efatmaneshnik and reidsema (2008).

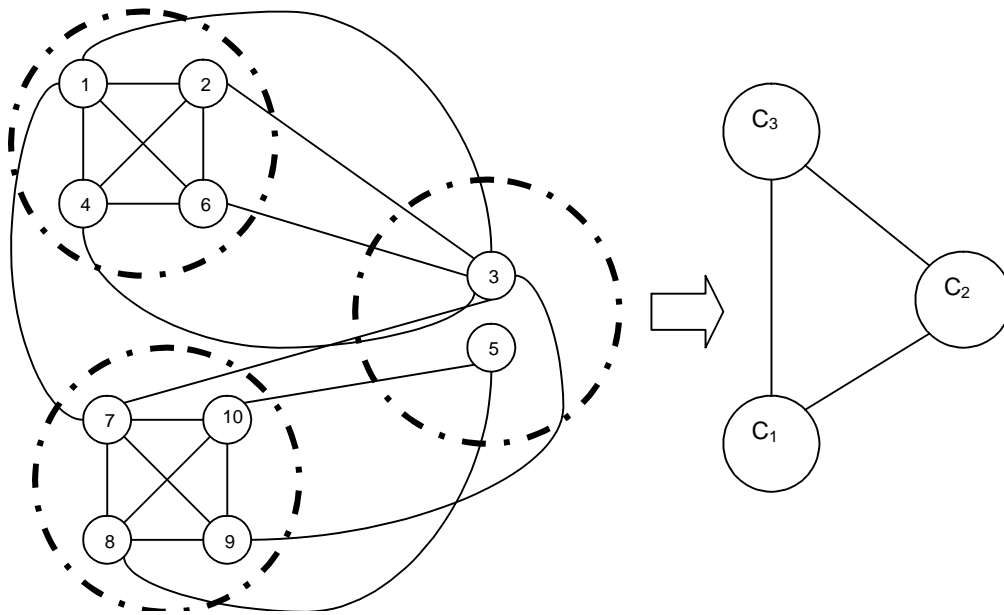


Figure 5. The partitioning and block diagram of graphs

## Strong and Weak Emergence

Reductionism is an approach to understanding the nature of complex things by reducing them to the interactions of their parts, or to simpler or more fundamental things. Adaptation of decomposition as a means of reducing the complexity of the main problem to several sub-problems is a reductionistic approach and can only be used for systems with weak emergence (since the system is essentially reducible). For systems with strong emergence decomposition is not a useful technique and instead holistic problem solving techniques must be used. Holism emphasizes the study of complex systems as wholes. One such design methodology is Enlightened Engineering developed by Bar Yam (2004) that is an evolutionary process. The most important and straight forward advantage of real complexity concept is that it can be used to determine whether a system has weak or strong emergence. In weak Emergence the emergent property is reducible to its individual constituents. In other words in such systems the whole is less than sum of the parts (in complexity). Since real complexity represents the complexity of the whole the latter can be written as:

$$C(B) \leq \sum_{i=1}^k C(P_i) \quad (5)$$

In which  $P=\{P_1, P_2, \dots, P_k\}$  are  $k$  partitions (parts or subsystems) of the system and  $C(P_i)$  are complexity of parts. Under such circumstances as (5), the complexity of the whole can be reduced to the complexity of the individual components. On the contrary a system has strong emergence when the emergent property is irreducible to its individual constituents and as such the whole is more than sum of the parts. In other words for systems with strong emergence a decomposition  $P=\{P_1, P_2, \dots, P_k\}$  of system that satisfy (5) can not be found.

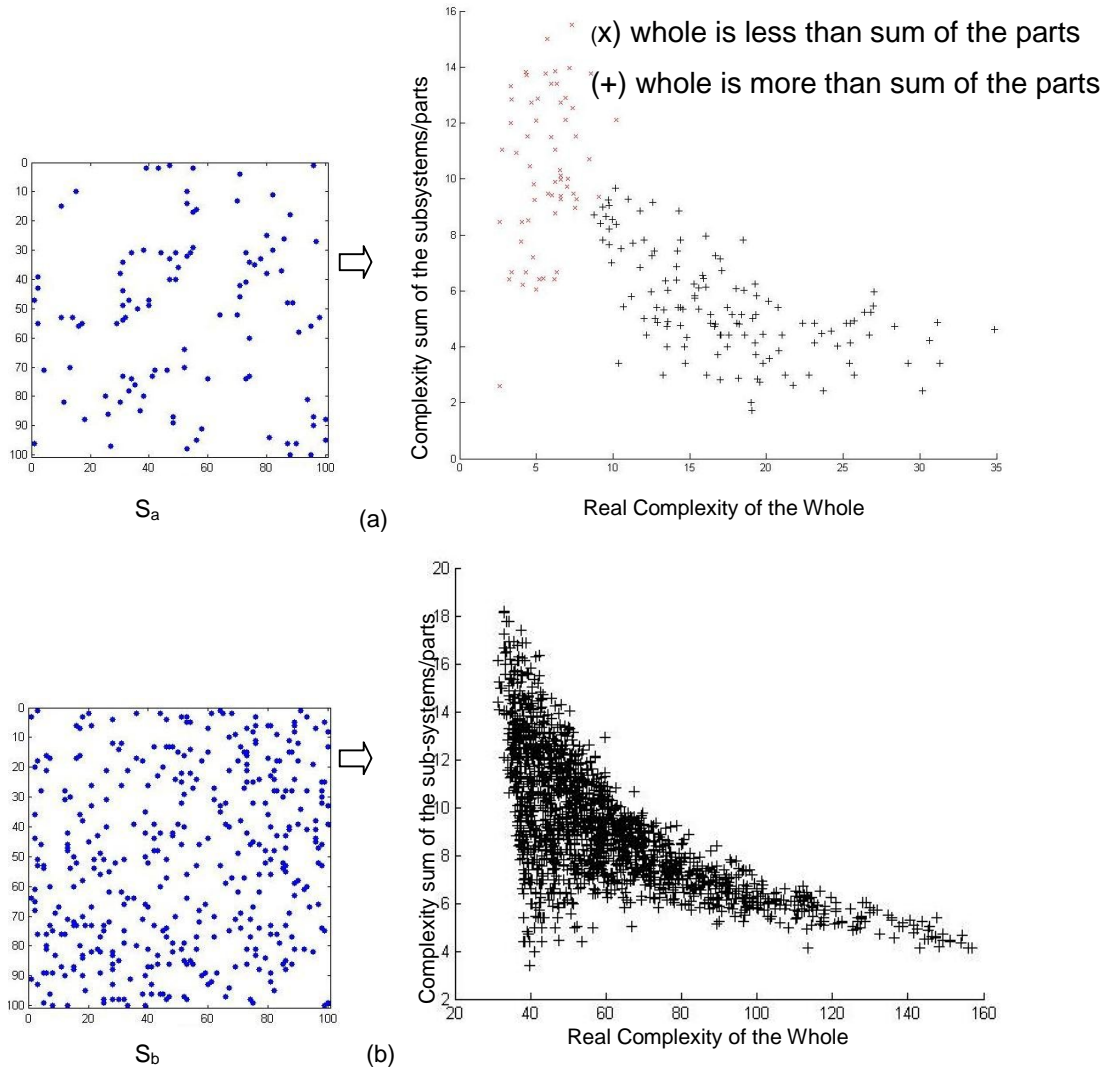


Figure 6. Determining weak and strong emergence in systems by using real complexity.

Obviously determination of kind of emergence in systems in the way presented here would be effected by type of base complexity measure used. In figure 6 by using Ontix as measure of complexity two systems are decomposed in many different ways (different subsystems). The two left matrices are PDSM (or self) of systems  $S_a$  and  $S_b$  in which each dot represents a link between the two nodes or variables. The system  $S_a$  in figure 6.a is of order 100 and size 100 where as system  $S_b$  (figure 6.b) is of the same order and size 400; system  $S_b$  is four times denser than  $S_a$ . Some decompositions of the system  $S_a$  do not show the whole (real

complexity) being more than sum (complexity) of the parts. Thus system  $S_a$  has weak emergence property since it can be reduced to the sum of its parts. On the contrary regardless of how system  $S_b$  may be decomposed it cannot be reduced to the sum of its parts and therefore system  $S_b$  has strong emergence. Figure 7 compares the whole and sum of the parts for many decompositions of the system presented in table 1. This system was decomposed in 200 different ways all of which had relatively balanced subsystems. The system showed strong emergence under all the viable decompositions.

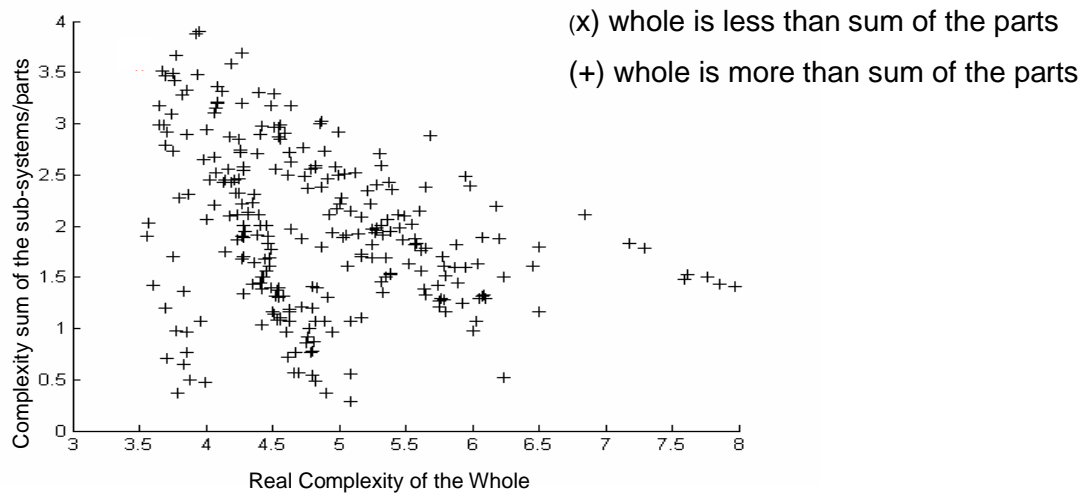


Figure 7. The example PDSM has strong emergence under all viable decompositions.

## Design Integration Schemes

Integration in design process can be carried out by using two major methods:

1. Supervised integration
2. Unsupervised integration

Supervised problem solving architecture involves high level integration teams and centralized planning (figure 8). According to Eppinger (1997), one important level of integration takes place within each development team; this is the now common practice of concurrent engineering, in which a cross-functional team addresses the many design and production concerns simultaneously. To assure that the entire system works together, sub-system development teams must work together and for that additional teams are assigned the special challenge of integrating those subsystems into the overall system (Eppinger, 1997). However for densely coupled problems/systems using high level integration teams as coordinator cannot be effective, since loaded coordination complexity would be a greater barrier to effectiveness of integration process.

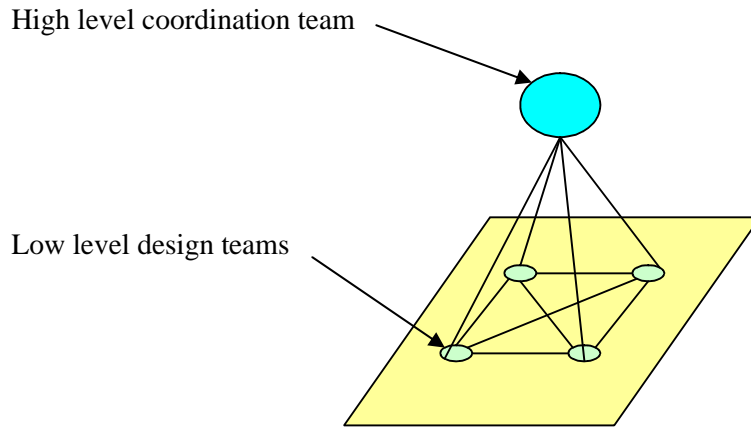


Figure 8. Integration team acts as a high level coordinator.

Unsupervised problem solving architectures can cope with problems of more complexity using distributed planning approach. These include systems that use 1) low level integration team and 2) multi agent architecture and 3) information intensive architecture. Systems using low level integrators and multi agent architectures correspond to two decomposition patterns that are recognized by Sosa et al (2000) as coordination-based and modular (figure 9). Coordination based decompositions partition the system into several relatively independent subsystems and only one (or few) severely connected subsystem(s) namely the coordination block(s) (figure 10). The identification of coordination block (figure 10.b) in a system can be performed through integer programming (Sosa et al., 2000). The coordination block ( $C_C$ ) is an integrative subsystem and the design team in charge of design of integrative subsystem is regarded as low level integration team that implicitly coordinates the activities of other teams. Since the design of the integrative subsystem must be much more complex than the other subsystems, the integration of the complex systems with more than a certain amount of coupling is not desirable by coordination based decomposition and low level integration scheme.

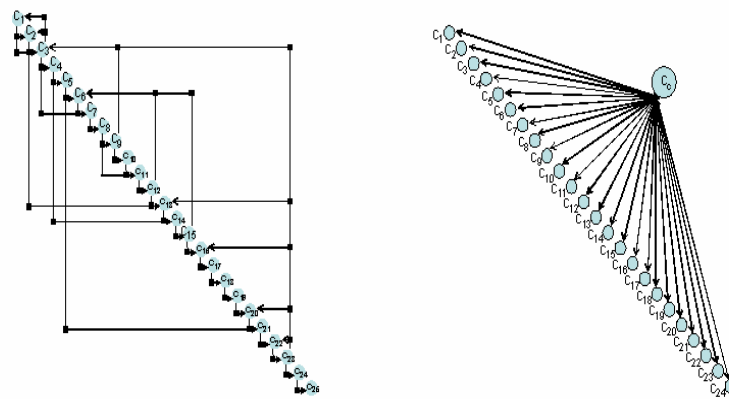


Figure 9. (a) Modular decomposition and (b) coordination based decomposition.

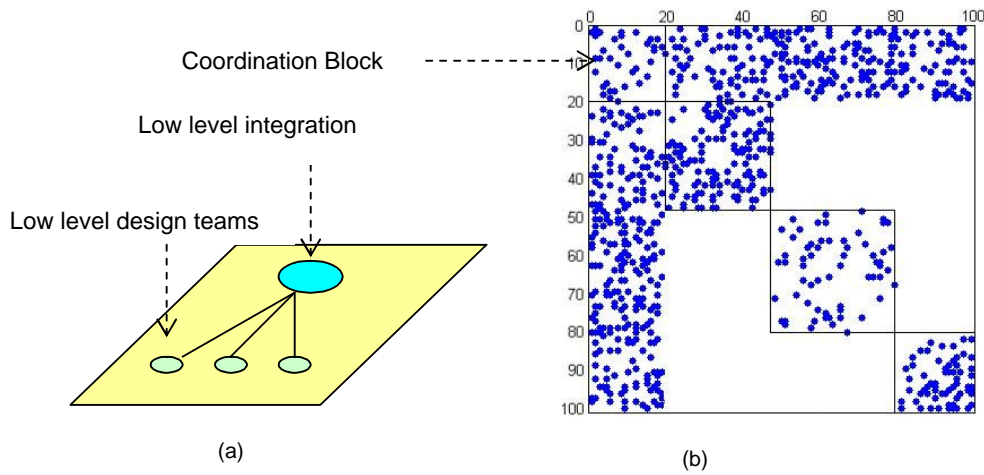


Figure 10. Interactions between design teams of low level integration scheme (a) and the corresponding PDSM of order 100 with coordination based 4-partitioning (b).

The interaction between the design teams in multi agent systems are autonomous and based on agents social knowledge. Multi agent systems are relatively complex field of research. The solution to the design problems in multi agent systems is formed in a self organizing fashion that emerges as result of autonomous interaction of the agents (figure 11.a); multi agent systems respond to modular problem decomposition (figure 11.b).

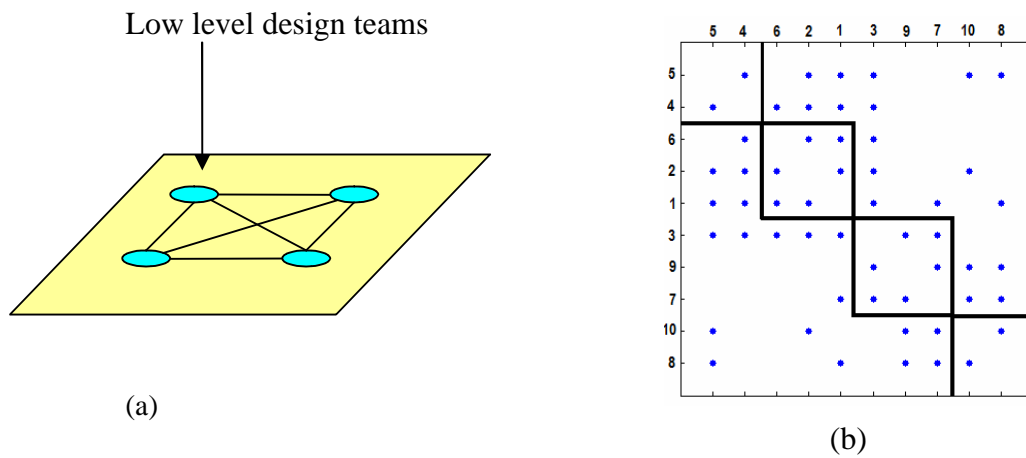


Figure 11. Multi agent design system (a), the corresponding modular PDSM decomposition (b).

Information intensive architecture can also be regarded as multi agent system in which the design teams (or coalition of agents) have overlapped and fuzzy boundaries. Information intensive architecture corresponds to overlap decomposition of product/system in which subsystems are overlapped and share some of the design variables with each other. The aligned organizational architecture constitutes coalitions that explicitly share some of the design agents (figure 12). The real complexity can be measured for overlap decompositions (Efatmaneshnik and Reidsema 2007). The main characteristic of this process model is the intense collaboration between coalition of agents making this mode an information and knowledge intensive process (Klein et al., 2003). As the impact of new information on the design process is relatively high, overlap decomposition mode and its corresponding integration scheme are suitable for problems of high complexity and self connectivity.

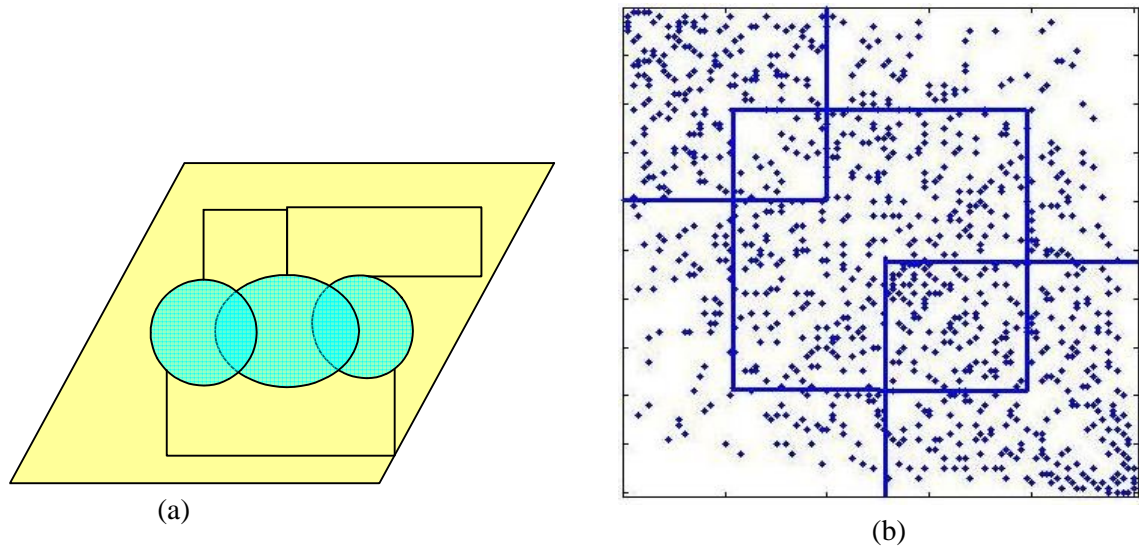


Figure 12. Design teams (a) as well as product partitions (b) have overlapped boundaries.

For really complex problems with high self connectivity, Bar Yam (2004) suggested that decomposition can not alleviate the overall complexity of the problem. He proposed Enlightened Engineering as an integration scheme for complex systems. Enlightened engineering is an evolutionary process of design that relies on radically innovative solutions provided by several design groups working parallel to each other on the same problem (Bar Yam, 2004). The evolutionary process is based on design teams competing for higher fitness of the solutions. Examining many design alternatives and avoiding premature commitment to routine solutions, innovative design groups have the ability to reach higher global optima and robustness: the solutions are tested for wider adoption to other solutions from other design teams (Bar Yam, 2003). We propose utilization of this integration scheme when no decomposition of the system leads to real complexity (whole) being less than sum (complexity) of the subsystems (parts).

## Conclusion

Decomposition and integration of complex problems was speculated. Integrated problem solving for complex problems requires a priori knowledge of system elements (variables) interactions. The interactions can be obtained from past products knowledge, expert idea, design of experiments and simulation techniques. The real complexity of the problems/systems after decomposition was introduced from the knowledge gained from PDSM of product. Real complexity directly points to integration difficulty of product/systems after decomposition. Real complexity can be used in reasoning about the type of integration scheme that most suits the product's level of complexity (figure 13).



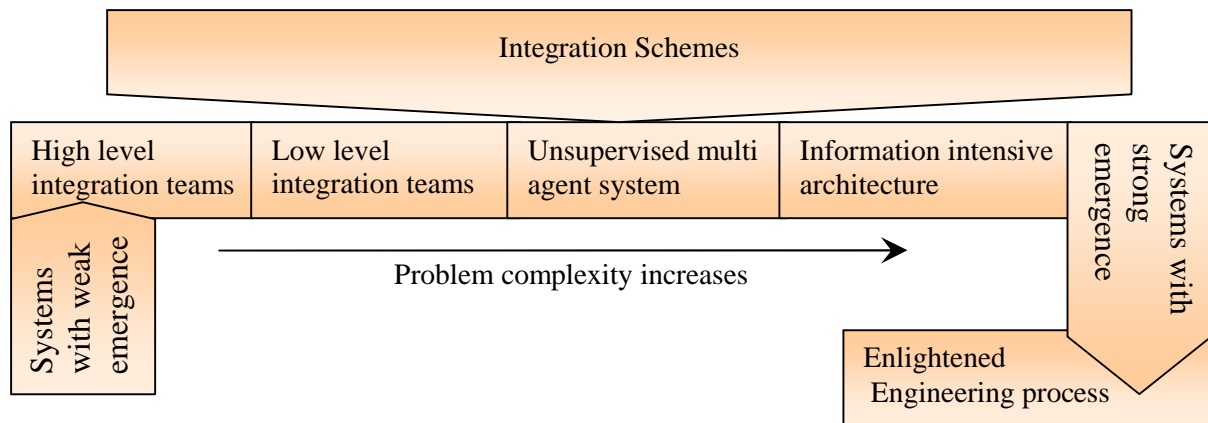


Figure 13. Ranking various integration schemes capability in coping with complexity.

## References

- Alexander, Christopher. 1964. *Notes on the Synthesis of Form*. Harvard University Press, Cambridge.
- Bar-Yam, Yaneer. 2003. When Systems Engineering Fails --- toward Complex Systems Engineering. Paper presented at the International Conference on Systems, Man & Cybernetics, Piscataway, NJ.
- Bar-Yam Yaneer. 2005. About Engineering Complex Systems: Multiscale Analysis and Evolutionary Engineering. In *Engineering Self-Organising Systems*. Edited by Sven A Brueckner. 16-31.
- Browning, Tyson. 1999. Designing system development projects for organizational integration. *Systems Engineering* 2: 217-225.
- Browning, Tyson. 2001. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management* 48: 292-306.
- Deshmuk, A.V., Talavage, J.J., and Barash, M.M. 1998. Complexity in manufacturing system, part 1: Analysis of static complexity, *IIE Transactions* 30: 645-655.
- Duin, R.P.W., and Pekalska, E., Object Representation, Sample Size and Dataset Complexity, In *Data Complexity in Pattern Recognition* (ed. by Basu, M. and Ho, T.K.), Springer, 2006, 25-47.
- Efatmaneshnik M, and Reidsema CA. 2007. Immunity and Information Sensitivity of Complex Product Design Process in Overlap Decomposition. In *Proceedings of 7<sup>th</sup> ICCS*, Minai A, Braha D, Bar-Yam Y (eds.) Boston, MA.
- Eppinger, Steven D. 1997. A Planning Method for Integration of Large Scale Engineering Systems. Paper presented at the International Conference on Engineering Design ICED 97, Tampere, Finland, August, 19-21.



Frizelle, G., Wu, Y., Ayral, L., Marsein, J., Merwe, E.V.d., and Zhou, D. 2002. A Simulation Study on Supply Chain Complexity in Manufacturing Industry, Manufacturing complexity network Conference, Cambridge, UK.

Guenov, M.D. 2002. Complexity and Cost Effectiveness Measures for System Design, Manufacturing complexity network Conference, Cambridge, UK.

Henderson, Rebecca M. and Kim B. Clark. 1990. Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. *Administrative Science Quarterly* 35(1): 9-30.

Holttä, K.M.M., and Otto, K.N. 2003. Incorporating Design Complexity Measures in Architectural Assessment, Engineering and Technical Design Conference, Chicago, USA.

Klein, M.,H. Sayama, P. Faratin, Bar Yam, Y. 2003. the Dynamics of Collaborative Design: Insights from Complex Systems and Negotiation Research. *Concurrent Engineering Research & Applications*.11(3): 201-209.

Klir, George J. 2003. Facets of Generalized Uncertainty-Based Information. In *Entropy Measures, Maximum Entropy Principle and Emerging Applications*. Edited by J. Karmeshu, Springer.

Kusiak, Andrew. 1999. *Engineering Design: Products, Processes, and Systems*. Academic Press.

Marczyk, J., Deshpande, B. 2006. Measuring and Tracking Complexity in Science, In Sixth International Conference on Complex Systems (ed. by Minai, A., Braha, D., Bar-Yam, Y.), Boston, MA.

Maturana, F., Shen, W., Norrie, D.H. 1999. MetaMorph: an adaptive agent-based architecture for intelligent manufacturing. *International Journal of Production Research* 37: 2159 - 2173.

McCord, Kent R. and Steven D. Eppinger. 1993. Managing the Integration Problem in Concurrent Engineering. Massachusetts Institute of Technology Sloan School of Management Working Paper 3594-93-MSA.

Michelena, N.F., and P.Y. Papalambros. 1997. A Hypergraph Framework for Optimal Model-Based Decomposition of Design Problems, *Computational Optimization and Applications* 8: 173-96.

Papalambros, P.Y. 2002. The Optimization Paradigm in Engineering Design: Promises and Challenges. *Computer-Aided Design*. 34: 939-51.

Pimmler, Thomas and Eppinger, Steven. 1994. Integration Analysis of Product Decomposition. ASME Design Theory and Methodology Conference Minneapolis, MN, September.

Prasad, B. 1996. *Concurrent Engineering Fundamentals, Volume II: Integrated Product Development*. Prentice Hall.

Simon, Herbert A. 1981. *The Sciences of the Artificial*, 2nd edition, MIT Press, Cambridge, MA.

Sosa, Manuel E., Steven D. Eppinger, and Craig M. Rowles. 2000. Designing Modular and Integrative Systems. Paper presented at International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Baltimore, Maryland, September.

Suh, N.P. 2005. *Complexity theory and applications*. New York.

Tanaka, K., Higashiyama, M., and Ohsuga, S. 2000. Problem Decomposition and Multi-agent System Creation for Distributed Problem Solving. Proceedings of the 12th International Symposium on Foundations of Intelligent Systems, Springer-Verlag.

Ulrich, Karl T., and Steven D. Eppinger. 2004. *Product Design and Development*, Third ed., Mc Graw-Hill/Irwin.

Yu, S.B., Efstathiou, J., 2002. An Introduction of Network Complexity. Manufacturing Complexity Network Conference, Cambridge, UK.

Zdrahal, Z., and Motta, E. 1996. Case-Based Problem Solving Methods for Parametric Design Tasks. Proceedings of the Third European Workshop on Advances in Case-Based Reasoning. 473-486.