

**Proceedings of DETC '08**  
**ASME 2008 International Design Engineering Technical Conferences and**  
**20th International Conference on Design Theory and Methodology (DTM)**  
**New York City, NY, USA, August 3-6, 2008**

**DETC2008-49361**

**A Multi-Domain Analysis Framework for Product Development**

**Joe A. Bradley<sup>+</sup> & Ali A. Yassine**

Product Development Research Laboratory  
Department of Industrial and Enterprise Systems Engineering  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801

**ABSTRACT**

The 21<sup>st</sup> century brings many new challenges to the product development (PD) community mainly due to a drastic increase in the scale and complexity of engineered systems. This requires the collaboration of various entities and resources within and outside firm boundaries. To address these new challenges, this paper proposes a novel framework for an enterprise-wide PD information management system.

The proposed framework provides an integrative view of the various dependencies and information flows that co-exist in three main PD analysis domains (i.e., people, products, and processes) and analysis methods for the discovery of gaps or 'misalignment' between them. These gaps could help explain why some organizations are able to provide more competitive products within a given industry. Moreover, the framework suggests that the characteristics of how an organization acquire data, interpret information, and apply knowledge will impact the final architecture of the product.

Finally, we "work-out" this framework by analyzing an open source software project, which offers some insights and new directions into how the transfer of data, information, and knowledge impacts the final (source code) architecture and design.

*(Keywords: Product Development, Design Decomposition, Product Architecture, Information Integration, Network Analysis).*

**1. A NEW PARADIGM FOR 21<sup>st</sup> Century PD**

Product development in the 21<sup>st</sup> century involves the development of large, multidisciplinary, and networked systems that cannot be embraced by a single group or organization. Decomposition, dispersion, and collaboration epitomize this new environment. Historically, the complexity

of engineered systems was gauged solely on technical considerations, where the use of system/product decomposition principles to manage such complexity has served the design community well thus far (Simon, 1969; Alexander, 1964; Rechtin and Maier, 1997). However, PD in the 21<sup>st</sup> century is an amalgam of product/people/processes and the information that flows between them resulting in a complex socio-technical system as depicted in Figure 1. This intricate system presents new managerial challenges to the engineering design community (King and Majchrzak, 1996; Collier, 1997).

The three domains, shown in Figure 1, are strongly related and glued together through various dependencies and information flows. After all, the development organization is executing the development process, which is implementing the product architecture. There is ample anecdotal evidence suggesting the existence of such relationships and possible 'misalignment' (or discrepancy) between them (Morelli et al. 1995; Hellgren and Stjernberg 1995; Eppinger and Salminen 2001, Sosa et al. 2004; Parashar and Bloebaum 2005). These observations are summarized as follows (note the areas numbered in Figure 1):

**Area 1:** The organization is structured to "efficiently and effectively" deliver on a specific design (i.e. product architecture) (Sanchez and Mahoney 1996). Examples include how GM organizes to produce automobiles, which is different than how Boeing organizes to produce jets. Of course, one can also argue that the existing organizational structure dictates (or at least facilitates or impedes) the kinds of product architectures possible.

**Area 2:** Once product architecture is selected, a project plan (or a process) is established to deliver the required transformations necessary for the product to

---

<sup>+</sup> Corresponding Author: jabradly@uiuc.edu.

materialize (Allen, 1997). Again, one can argue that the process in place influences the product architecture choice.

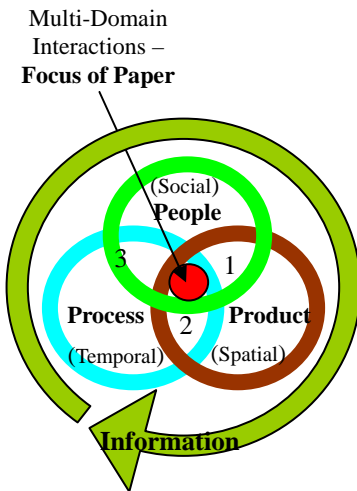


Figure 1: 21<sup>st</sup> Century PD

**Area 3:** The third area is often less obvious, but it is interpreted as the relationship between the process (e.g. project plan) and the organizational structure. A process requiring frequent bi-directional communication among various development teams must be supported by the existing social network of the development organization (Levitt et al. 1999). Moreover, this social network itself is evolving in response to different project plans that may have lasting implications (beyond the finish of the development project) on the social characteristics of the development organization.

Design decomposition creates serious managerial challenges due to the complex web of interdependencies *within* and *between* the decomposed domains. Isolated management procedures result in integration problems that manifest themselves in excessive design iterations, design churn, organizational memory lapses (regarding design problem solving know-how), and deteriorated morale amongst designers. These are not hypothetical observations, but have been widely reported in various industrial settings including automotive (Yassine et al. 2003a), aerospace (Browning et al. 2002), construction (Ford and Sterman 2003), electronics (Wheelwright and Clark 1992), and software (Abdel-Hamid 1988). There are few rigorous models that help mitigate these risks and there is a pressing need by the design community for an enterprise-wide model of design decomposition management (Collier, 1997).

The risks associated with multiple domain decompositions must be well defined in order to develop useful PD management models and risk mitigation strategies. The risks of decomposition in 21<sup>st</sup> century PD are outlined next.

**Integration risk:** Integration is the process of aggregating the decomposed pieces (or product modules) back together to deliver the final product design (or system). Carefully decomposed products have ‘clean’ (i.e. well-defined) module boundaries and interfaces which will facilitate system testing and integration. On the other hand, poor product decomposition translates into overlapped responsibilities (among module owners) and deteriorated final product performance (Yu et al., 2007).

**Iteration risk:** The interdependence between design and development tasks makes engineering design fundamentally iterative, driven by the repetition (rework) of tasks due to the availability of new information (generated by other tasks) such as changes in input, updates of shared assumptions or the discovery of errors. As this uncertain information becomes available, the tasks are repeated to either verify an initial estimate/guess or to meet design specifications (Meier et al., 2007).

**Churn behavior:** Iteration not only slows down design convergence, but also may have a destabilizing effect on the system’s behavior (Yassine et al., 2003a). It is not hard to see that this interdependence opens up the possibility of activities being inter-linked in a pernicious way that results in a “chain-reaction” of updates in the form of new information “flooding” the system. In these pathological situations, most of the developmental effort of the team will be spent in updating plans-of-action, only to have them changed later, and no real progress is made (i.e. design churn).

**Locus of innovation and decision-making:** Another important consequence of decomposition is the distribution of the locus of innovation and decision-making throughout the decomposed (but interlinked) development network. As such, no single designer/team has full knowledge of the overall system and identifying potential sources of architectural/system innovation becomes difficult; unless careful analysis techniques are employed (Batallas and Yassine, 2006).

**Misalignment risk:** The three decompositions simultaneously coexist: product decomposition into modules, process decomposition into design tasks, and organizational decompositions into design teams. If these decompositions are not well coordinated across the various domains, then inefficiencies may arise (process, product, and/or organizational inefficiencies) (Sosa, 2008).

In this paper, we propose a framework that illustrates a multi-domain approach that integrates the design team, the artifact, and a characterization of the information shared amongst the design team members. Thus, developing the basis for an integrative, enterprise-wide PD information management system that is capable of addressing the challenges of 21<sup>st</sup> century PD, described above.

The framework we propose considers a Data → Information → Knowledge → Product schema. This model attempts to provide additional insight into the argument that products design organizations as well as the argument that

organizations design products.<sup>1</sup> This framework suggests that the characteristics of how an organization acquire data, interpret information, and apply knowledge will impact the final architecture of the product. Consequently, if its industrial environment changes and the current organizational structure are not competitive, it would be important to understand the effect of changes to either the organization or the product (Henderson and Clark, 1990).

The rest of the paper proceeds as follows. In the next section, we provide a through literature review of publications discussing many analyses, theories and results considering a multi-domain approach to product development. In section 3, we introduce the Data → Information → Knowledge → Product framework schema for studying the flow and transformation of data in the PD network. Additionally, Section 3 proposes several constructs to operationalize the framework and “workout” some of the framework computational details. Section 4 discusses a validation plan for the proposed framework using an open source software example. Finally in section 5, we summarize the efforts of this research project and the research plans moving forward.

## 2. LITERATURE REVIEW

Product development (PD) of complex products is typically decomposed into inter-related groups performing numerous tasks in order to successfully develop a product that satisfies a need. Consequently, product development is a multi-faceted endeavor requiring the choreographed relationship of many domains. Eppinger and Salminen (2001) discuss the possible relationship between DSMs for the physical, tasks, and organization domains. Sharman et al. (2002) suggest that elements in one domain need to map to the same element in another domain in a one-to-one manner. They propose a hypothetical optimization of a multiple-domain PD project resulting in an optimal DSM showing the relational arrangement of elements in the various domains. Danilovic and Browning (2007) propose a rectangular DSM construction relating DSM’s representing different domains of the product development process. Such construction is necessary because the intersection or union of two domains need not be the same exact size or in the case of the DSM’s the same dimension. This new domain mapping matrix (DMM) provides insights into the various characteristics of the product development process. It is worth noting that matrix mappings approach, in general, between various interrelated domains was discussed thoroughly in Yassine et al. (2003b).

Maurer et al. considers a multi-domain approach that considers the complexity cycle for multiple factors including market complexity, product complexity, process complexity, and organizational complexity. They propose a scheme that relates these domains by elements of information sharing

activity taking place within the organization. These multiple domain elements map to a new multi-domain network. Gokpinar et al. studies the relationship between the product architecture and the organizational structure by considering their joint impact on efficiency and quality in complex product development. They found that coordination problems, within the development team, have a direct positive correlation on the overall quality of the product. It would be interesting to know if the coordination issues lead to architectural problems downstream. Along similar lines, Sosa et al. (2007) considers a mapping between a product, organization, and process in product design. He incorporates an affiliation network to measure a participants overall involvement in the design. He suggests that this mapping scheme will allow organizational managers to quickly identify “whom” information or iteration design requirements should pass through. Considering only the organizational architecture, Krackhardt and Carley (1998) propose a network-based approach in terms of three domain elements – individuals, tasks, and resources. Based on the relational matrices it is possible to use general matrix multiplication techniques to build new multiple domain aggregated information. This provides a useful technique to understand the work flow patterns and processes. For example, aggregating a task and resource networks, we can gain some understand of which task require which resources as well as which resources are over- or under- utilized. Shooter et al. (2000) proposes a model of the information flow by considering data in various forms. They propose that design activities operate on information, which is basically a description of the product being design.

The effort in these studies is to understand or decipher the relationship between the product, the organization, and other possible factors. We can characterize these as product architecture, organization architecture, and information or content architecture. From literature it would appear that researchers predominantly view the relationship between product architecture and organizational architecture as starting with the product architecture and then working out an organization design or architecture. For example Baldwin and Clark (2000) derives the idea of a task structure matrix (TSM) by starting from the product component DSM and mapping out required organizational tasks needed to realize the desired end-product. However, other researchers suggest that products design organizations. Sanchez and Mahoney (1996) consider strategic decision-making in a firm and how the organizational structure influences the available options of product architectural characteristics. Sako (2003) suggests that the product architecture gives greater scope in the choice of organizational design. Thus given multiple organizational designs for a single product architectural choice, it is possible to trace the patterns of information flow to help determine the best organizational design among various options. However in contrast, organizational inertia may have a stronger influence on the available options for the product architecture. These

---

<sup>1</sup> Perhaps, over time, one argument or perspective becomes stronger within an organization.

studies suggest that the product architectural choice of a firm influences the organizational design; however, pre-existing organizational structure and capabilities impacts product design (Gulati and Eppinger, 1996). Consequently, there is a two-way relationship between product and organizational architecture. For example, Sako (2003) points to the fact that IBM adopted a non-modular organizational structure to manage its System 360 mainframe computer which is a modular architecture. IBM desired to keep the module interfaces proprietary to prevent others from developing compatible modules. Thus, the desire to manage modular interfaces influenced the organizational structure. Furthermore, there is evidence in the disk drive industry that suggests that the organizational architecture was a key influence in the overall product architecture. Chesbrough and Kusunoki (2001) found that certain firms had high-level of expertise and specialization linked to within-module innovations with the disk drive industry. Consequently these firms had no incentive for changing the product architecture. Any changes in the architecture could make their expertise obsolete. This example suggests that the product architecture could be constrained by the organizational architecture. Now revisiting a definition of product and organizational architecture (Ulrich, 1995; Ulrich and Eppinger, 2002; Sako, 2003,) as a bundle of characteristics that define (1) interfaces between elements of the whole, (2) a function-to-component (or task-to-organizational unit) mapping that defines the elements, and (3) the decomposition of the whole into components, functions, rules, and/or tasks; then as stated earlier, multiple dimensions and domains exist and must be considered in defining the overall product development process. Consequently, choosing an optimal decomposition or specifying an optimal component or module boundaries is non-trivial. Iyer et al. (2004) proposes a model that integrates process, information/knowledge, infrastructure and organization domain. This model proposes a product architecture design method that takes into account this multi-domain approach. They suggest that this approach can provide a better assessment of the *emergent* architecture for a given technical project.

Although there is a body of literature that considers the links between the many different architectural domains (product, organization, task, etc.) in the product development process; what seems to be missing is a relational analysis and investigation of the mediating element linking the product and organizational architecture. From the studies discussed above it is clear that a simple decomposition of a product into its many sub-assemblies or functional components will not necessarily have a one to one matching or decomposition to an organizational or task structure. Furthermore even when such a decomposition is worked-out the resulting patterns of information or relational network show some evidence of misalignment (Sosa et al., 2004; Bradley and Yassine, 2006). Consequently there must be additional factors to consider in understanding the relationship between the product and the

organization design. Sako (2003) considers a mix of strategic drivers that appears to mediate the relationship between the product architecture and the organizational architecture. Sako suggests that marketing decisions (i.e. build to order), manufacturing decisions (subassembly production lines), and financial investment decisions (reduction in assets via outsourcing) are factors in understanding the relationship between the product and organizational architectures. Additional Galbraith (1977) suggests that organizational design is a decision process that is entrenched in overarching goals of the firms and its purpose for existing. Furthermore, the division of labor (functional units) and the coordination linkage determines the organizational communication channels. Conway (1968) suggests that the communication patterns in the organization determine the final “system design” (i.e. product architecture).

### 3. THE PROPOSED FRAMEWORK

One way to think of a product is the reduction of information and knowledge into a usable form (i.e. products and/or services). To reduce this information to a usable form, an organization must successfully (1) collect *data*, (2) process data to extract new *information*, (3) combine this information to generate new *knowledge* and *insights*, and (4) use this new knowledge to develop *products* and *services* that satisfies a customer need. A fundamental challenge is to understand just how an organization traverses from data collection to a final product. We propose that these relationships can be elucidated via network relational analysis.

The network analysis that we are using in this paper borrows many of its mathematical concepts from graph theory and social network analysis (Harary, 1969; Wasserman et al., 1994). The benefit of network analysis, when analyzing complex systems, is that it considers not only the influence of the individual elements but also the relationship amongst them. In the case of product development, we consider three networks, a team network (individual design teams and groups) a physical network (components and subsystems), and a content network (data, information, and knowledge). We can construct a graph defined by nodes (actors or development participants) and edges (relationships between actors) to represent a PD network, see Figure 2. If the graph of Figure 2 represents a team network (i.e. organizational architecture), then nodes A through E are the development team participants and an edge between them represents a communication link (or any other relationship between them). On the other hand, if the graph represents a physical network (i.e. product architecture), then nodes A through E represent product components or subsystems and the edges represents a physical relationship (e.g. dependency or interface) between them (e.g., subsystem A interfaces with subsystem B).<sup>2</sup> Furthermore, we

---

<sup>2</sup> An additional PD network can be constructed from considering the development activities (i.e. a PD process network). This will lead to a project network similar to traditional project networks such as CPM and PERT networks.

can extend this to incorporate the idea of a content network (e.g., data, information, and knowledge). In this case, each node either represents data, information, or knowledge (heterogeneous nodes and links). These nodes represent distinctly different categories of information and the links represent relationships amongst the design team members based on how they source this information.

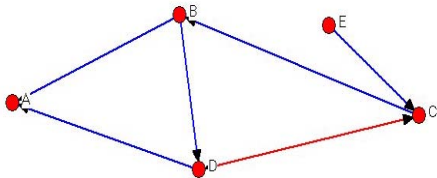


Figure 2: Represents a 5 node graph

Additionally, to illustrate the relation structure of our data we take advantage of the capabilities of the Design Structure Matrix (DSM) (Steward, 1981; Browning, 2001; Sharman and Yassine, 2004). The DSM is a useful tool for visualizing the relationship amongst a set of entities and analyzing several different situations depending on the type and nature of the information being exchanged. The organization interactions, component interfaces, and content transformations can, then, be arranged using a DSM representation, as shown in Figure 3. This representation allows us to specify the relational attribute within and between these domains.

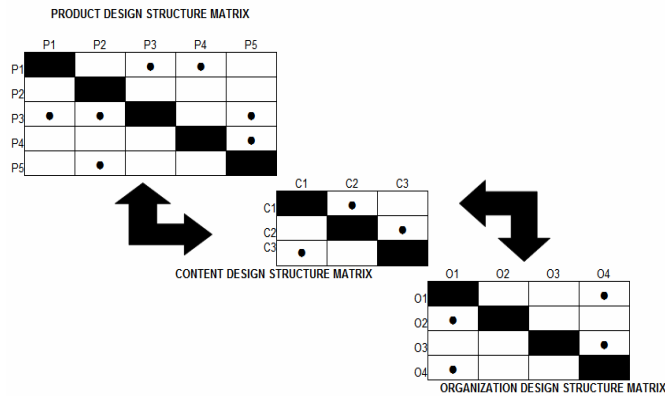


Figure 3: Multi-Domain Relational Model

### 3.1. Framework overview

An overview of the proposed framework is shown in Figure 4. In this framework we consider the product development process to be a set of multiple layers consisting of various transformations and rules to progress from one layer to the next. Our model slightly borrows from the idea of Data → Information → Knowledge → Wisdom (Ackoff, 1989; Bellinger et al., 2002). However their approach focuses on machine-learning and organizational change, respectively.

Alternatively, we study system aspects of the product development process and attempt to frame the movement between and among layers in new definitions.

The benefit of framing the product development process along a Data → Information → Knowledge → Product schema provides a level of abstraction that allows an organization to gain some insight into new ways of organizing; for example, instead of organizing around project, products, or functional units, it may be possible to organize around knowledge. Also, this may provide a mechanism by which an organization can identify gaps in its data acquisition, information interpretation, or knowledge application. These gaps could help explain why some organizations are able to provide more competitive products within a given industry.

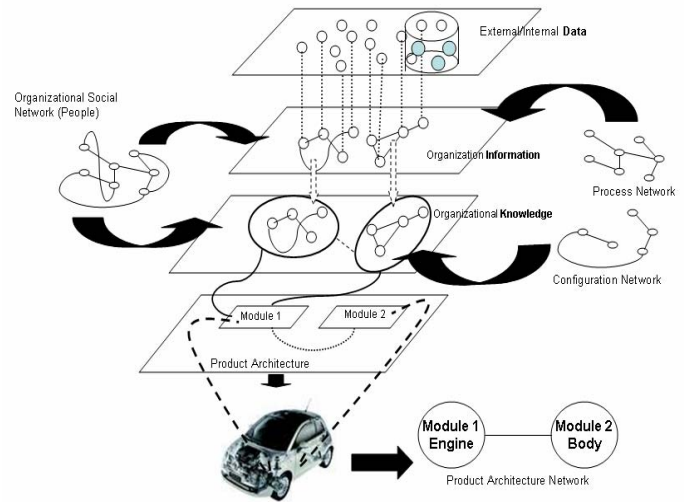


Figure 4: Data → Information → Knowledge → Product Schema

Furthermore, the proposed framework supports a re-engineering of the product development process. That is, the feed-forward process suggests that we follow these steps: data → information → knowledge → product. Consequently the feedback process would suggest that the reverse is possible: product → knowledge → information → data. Based on this, we can think of new ways to architect products and organizations in a feed-forward or feedback construct.

One of the key requirements for this framework (Figure 4) is to appropriately define and characterize each layer and the various transformations or relationships that occur between these layers. The definitions below will provide more detailed explanations.

### 3.2. Data layer

The data layer consists of incongruent, non-aggregate, disconnected elements. This can be considered as data collected independently among various functional units within an organization. We suggest that each of these functional units are engaged in some form of environment scanning that enables them to collect streams of data that is vital to their role

within the organization. Sources of data may include journals, industrial tradeshows, conferences, friends, individuals within the organization, professional contacts as well as many others (Aguilar, 1967). We claim that the data gather during the scanning efforts is not yet information - based on the definition used in this paper. We suggest that the data layer consists of databases authored and organized from either internal or eternal sources as in Figure 5.

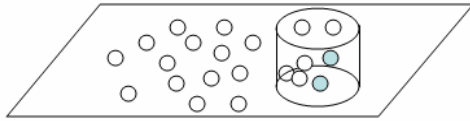


Figure 5: Databases and sources of internal and external Data

### 3.3. Organizational information layer

The information layer represents sets of connected elements of the data layer (i.e. external and internal data). This layer is the result of relationships among the people, in the disparate functional units (i.e. organizational network), attempting to answer the “who”, “what”, “when”, and “where” questions regarding some factors that impact their tasks. The links in this layer result from people in the social network sourcing various types of data from the data layer. As discussed in section 3.2, this data can be authored by individuals or groups whom are within or external to the organization. We can regard this as a composition of dyadic relations (Wasserman and Faust, 1994).

Therefore, when discussing the transition from the data layer to the information layer, the unit of analysis will be dyads in the network (see Figure 6). The justification for using a dyad is based on our definition of information -- information is data that is processed to be useful (Ackoff, 1989). We extend this definition to say that members in the organization will seek out one-on-one relationship to obtain or create useful information. We view this as strictly dyadic in our analysis; however, any one node in the network can have several dyadic relations. For example, a person that is a member of the marketing department may have a relationship with someone in both engineering and finance. This relationship could be due to one of three scenarios: (1) he/she sources data from engineering and finance databases, (2) he/she communicates directly, person-to-person, only within the social network, or (3) he/she sources data from databases as well as communicate person-to-person. Note that if an individual only sources data from a database without the person-to-person communication this link will not be obvious. Therefore, these individuals may not be aware of their impact on the overall PD process. This could result in miscommunication and/or misinformation which is critical for downstream integration. If we consider such a network, it will consist of three nodes and two links; however, we consider the dyadic relationships in this layer as shown in Figure 6.

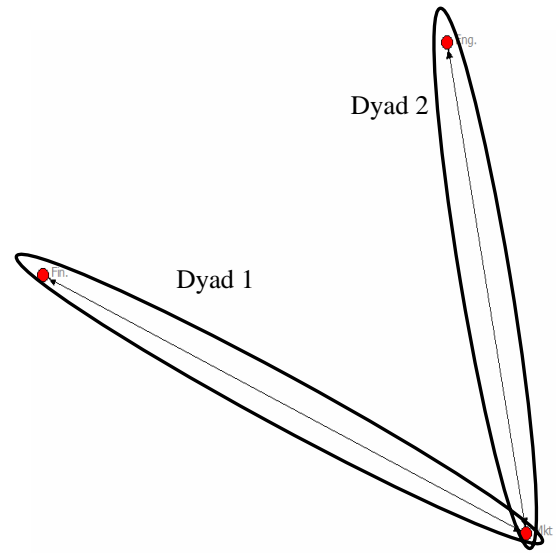


Figure 6: Dyads in the Information Layer Network

Using these dyadic relationships, we can compute a relational matrix for each team member (e.g. development collaborator) as shown in Figures 7 and 8. Figure 7, for example, shows the dyadic relationships between team member  $C_0$  and the remaining members in the network ( $C_1, C_2, C_3, C_4$ ) by noting if one of the other collaborator have sourced information from the same database ( $DB_1, \dots, DB_5$ ) authored by  $C_0$ . Note that in Figure 8 example, team members  $C_0$  and  $C_1$  have switched positions, since, in this instance, we are considering the expansiveness of the database files authored by  $C_1$ .

Database files authored by a single design team member,  $C_0$

	$DB_1$	$DB_2$	$DB_3$	$DB_4$	$DB_5$
$C_1$	1	1			1
$C_2$	1	1	1		1
$C_3$		1		1	1
$C_4$			1	1	1

$C_1 \dots C_4$  = represents team members  
 $DB_1 \dots DB_5$  = represents database files authored by a given team member,  $C_0$

Figure 7: Collaborator Relational Matrix for collaborator  $C_0$

Database files authored by a single design team member,  $C_1$

	$DB_1$	$DB_2$	$DB_3$	$DB_4$	$DB_5$
$C_0$	1	1			1
$C_2$	1	1	1		1
$C_3$		1		1	1
$C_4$			1	1	1

$C_0, C_2, C_3, C_4$  = represents team members  
 $DB_1 \dots DB_5$  = represents database files authored by a given team member,  $C_1$

Figure 8: Collaborator Relational Matrix for collaborator  $C_1$

The aggregation of the relation matrices can be used to identify sharing of data (i.e. CAD designs, design specifications). If a team member references or uses someone's design or specifications, this suggests that *new information* has been generated as a result of this dyadic interaction. As a measure of information, we propose a *collaboration index* (CI).

$$CI_j = \sum_{i=1}^m a_{ij} \quad \forall j \quad (1)$$

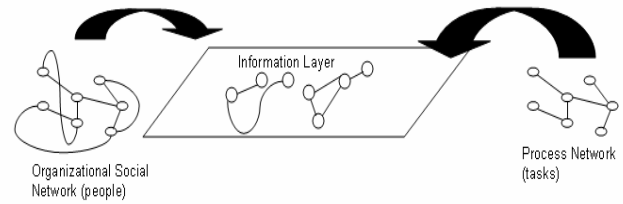
Where  $a_{ij}$  are the entries in a  $m \times j$  collaborator relational matrix,  $m$  is the number of design team members involved, and  $j$  is the number of databases sourced by other team members created by an individual or group of members. Additionally, to measure the impact of each member on the overall information content in the network, we propose a *collaboration impact value* (CV).

$$CV = \sum_{j=1}^p \frac{CI_j}{p} \quad (2)$$

Where  $CI_j$  are the collaboration indices for each team member in the project, and  $p$  is the total number of dyadic relations in the network. We hypothesize that these measures impact the organization's ability to aggregate information and subsequently create knowledge.

Operating on this layer are both the organizational social network as well as the task or process network, see Figure 9. For example given a listing of 5 tasks that must be completed for successful development of a product, each team member will source data from various sources to complete the tasks. The processes outlined in the process network for completing the task provide the mechanism by which data is converted or transitioned into information layer by way of creating linkages between individuals and/or teams in the network. The justification for why this relationship is considered information is based on our definition of information (an aggregate of useful data). We suggest that the process network provides tasks that must be completed (e.g. tasks that are necessary to design an automobile engine); subsequently each task is assigned to an individual or group (e.g., (1) piston design, (2) block design, (3) cooling design). We speculate that these individuals or groups will need to source databases to acquire needed data (i.e. the block designer(s) will need specifications about the piston). In this case, consider the scenario that the block designer does not communicate with the handle-bar designer(s) in person-to-person relationship (social network) but there linkage is based only on sourcing from databases. At this point information is created between the piston and the block designer(s), because the block designer(s) has utilized the specifications from the piston designer(s) to establish the correct path in proceeding to design the block to interface with the piston. It can be argued that a person-to-person link should be established

between the designers that could possibly result in a transfer of tacit information between the designers that could allow for better overall product architecture.



**Figure 9: Information Network of Personnel in Functional Units**

### 3.4. Organizational knowledge layer

Knowledge can further be defined as information with a high degree of certainty and validity (Braha, 2001). Thus, the knowledge layer represents a set of connected elements of aggregated blocks of information in the information layer. The knowledge layer is the result of the organization's attempt to answer the "how" and "how much" questions regarding a certain design decision. These questions are focused on how to realize the product or service that satisfies customer needs at an acceptable production or operational cost. We propose that the knowledge layer consists of network relationships beyond the dyad. In this framework, we consider knowledge to be a higher level of complexity; i.e., triadic, quad, etc. relationships<sup>3</sup>.

At the knowledge layer, the organization is in the process of applying the *data* and *information* to create *knowledge* to realize a desired result – product or service. For the development of a complex product, the expansiveness of a dyadic relationship is very limiting.<sup>4</sup> In this framework, we have considered dyadic relationships as pieces of information (i.e. useful data). However, as we transition to the knowledge layer, we have extended the unit of analysis to triadic. Of course, it can be argued to extend the unit of analysis to quads or larger sub-network. The justification for this approach is that as the path length between the network nodes increases, the network search for and the transfer of information becomes more expensive and complex (Hansen, 1999; Sorenson et al., 2006). Furthermore, if we accept the argument of bounded rationality (Simon, 1957) then the search space of human capacity will also have some limitation.

Because knowledge can be defined as an aggregate of information with a high degree of validity and certainty, we extend the concept of a triadic relationship to include the property of "transitivity". A definition of transitivity states – the triad involving actors  $i$ ,  $j$ , and  $k$  is transitive if whenever  $i \rightarrow j$  and  $j \rightarrow k$ , then there also exists  $i \rightarrow k$ . We speculate that transitivity is proxy for characteristics of repeatability,

<sup>3</sup> We consider dyads, triads, quads as network building blocks or motifs (Artzy-Randrup et al., 2004).

<sup>4</sup> Such complexity is illustrated by Sosa et al. (2004) in their analysis of a jet engine product development effort.

validity, and certainty in the information provided by a single member or group of members in a design team. In this framework information will not transition to the knowledge layer unless it is sourced or used frequently by others (i.e. high transitivity incidences). We hypothesize that a team member who has many designs, specifications, resources that exhibit high transitivity will have a high impact on knowledge formation. As a measure, we propose a *knowledge influence index (KI)*.

$$KI_j = \sum_{i=1}^q \frac{TF_{ij}}{q} \quad \forall j \quad (3)$$

where  $TF_{ij}$  is the total number of transitive relationships for a given design team member and  $q$  is the total number of transitive relationships in the network.

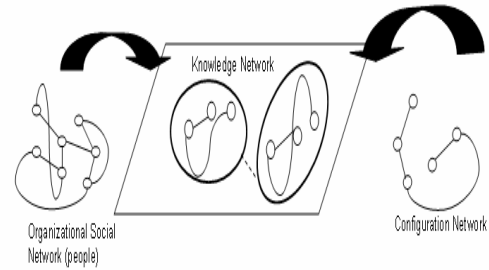
This index provides an indication of the overall impact of a team member on the patterns of knowledge formation in the network. Furthermore, it suggests high repeatability, validity, and certainty in the information provided by this team member. We suggest that a designer with a high *KI* value will have control on “how” things are done or placed into modules (i.e. architecture) for the PD project.

Since the knowledge layer consists of highly reliable information elements, we can speculate that this layer represent organizational “know-how”. For example, there exists organizational knowledge on how to successfully develop pistons for an automobile engine, or maybe the cooling system for an automobile engine. The challenge is now to successfully map this knowledge to functional units and subsequently reduce to modules for a final architecture. Acting upon this layer is the organizational social network and a configuration network (Ulrich & Eppinger, 2004), see Figure 10. We propose that for teams or individuals that have information elements that have transitioned to knowledge, a communication link between the entities must exist within the organizational social network. Because at the knowledge layer, the organization is pursuing the application of data to realize a product, it is essential that individuals and teams communicate if there is an informational relationship between them. We propose a search scheme within a configuration network analyzing various combinations of knowledge elements as a result of the communication link between the individuals or the teams. These individuals and teams discuss various options to help realize a product architecture. When successful combinations of knowledge elements and configurations have been found the elements are now decomposed into various chunks (product architecture layer).

### 3.5. Product architecture layer

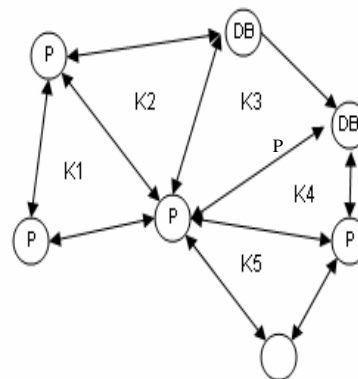
The product architecture layer represents a chunking of functional elements (knowledge elements) into an aggregate product by answering the “how-to-do-it” question by way of the reduction of knowledge into a tangible product or perhaps

a service. However, the mapping of the knowledge elements to functional chunks is non-trivial. For example in considering



**Figure 10: Knowledge Layer Networks**

a mechanical system or device, there exist some physical connectivity limitations (Whitney, 2004). Such limitations will determine the scope of architectures that can be realized. To illustrate this concept, consider Figure 11 which represents a knowledge network for an automobile design, comprising five knowledge elements (K1,K2,K3,K4,K5)<sup>5</sup>. For example, we can consider K1 and K2 to represent knowledge about piston design, K3 and K4 to represent knowledge about block design, and K5 to represent knowledge about body design. Now that the knowledge exists on how to design these different components, we can consider how to interface and connect these knowledge elements. Recall from section 3.4, that acting on the knowledge layer is a configuration network which determines possible architectural choices. We hypothesize that these architectural choices also depend on connectivity limitation and constraints. Therefore, we can reduce the knowledge network in Figure 11 to a triad sub-network where the links between the nodes represent interface constraints, as shown in Figure 12. Finally, Figure 13 shows a mapping to a final product by way of combining knowledge elements (Figure 12) into modules and subsequently into a physical artifact.



**Figure 11: Knowledge Network Representation**

<sup>5</sup> Recall that in the knowledge layer we have defined knowledge as triadic relationships (with a transitive property) of information elements based on the information layer.

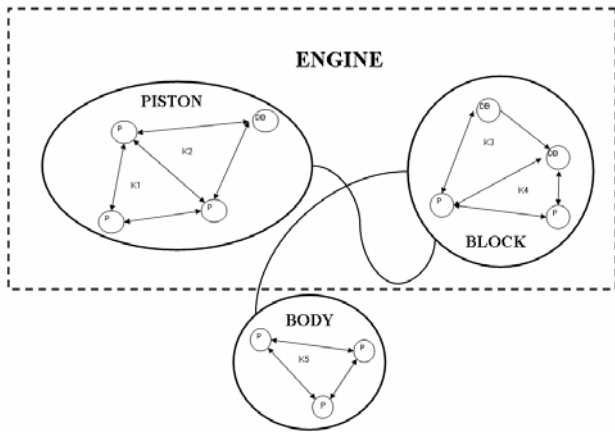


Figure 12: Knowledge Elements

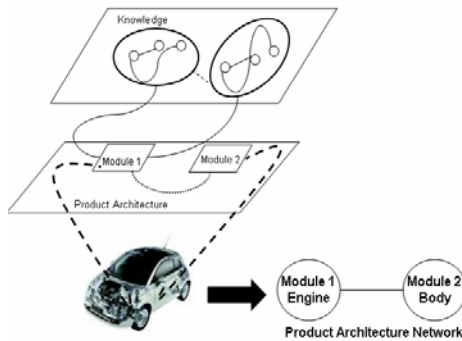


Figure 13: Mapping of Knowledge to Product

## 4. VALIDATION PLAN

### 4.1. Robocode example

Open source software projects provide an excellent opportunity to work through this framework. One reason is that open source software development (OSSD) exploits the distributed intelligence of participants across the world (Kogut, et al., 2001). Another reason is that open source can be described as a user-centric innovation (von Hippel, 2005).<sup>6</sup> If considered user-centric, then open source allows us to directly understand the customer requirements as well as how the project is decomposed into separate design teams. We can investigate how these known software requirements shape the communication patterns amongst the developers and subsequently impact the overall architecture of the software source code.

For our investigation we use the open source Robocode development project, which is an educational game (<http://robocode.sourceforge.net>). The game is designed to help teach java programming to gamers as well as to have fun competing with other robot designers and programmers. The

<sup>6</sup> By user-centric, we mean that an individual or a group initiated the development effort with the intent of developing a product that will be first and foremost beneficial to them (Comino et al., 2007; von Hippel, 2005).

Robocode software source has 18 package files and 181 class files. There are 27 project collaborators (programmers). We can construct the software source code network at both the package and class level. The relationship between the packages or the class files are based on function or method calls between the files. We can construct a binary DSM of this network by specifying “1” if a class or package makes a function call to either request data from or send data to another and specify “0” otherwise. The collaborator network (programmers) is based on who communicates with whom via chat log as well as who submit code updates and changes to the same source files (e.g. packages or class files). The collaborator network is 2-mode; it contains relational patterns when developers are playing the game as well when they are submitting code updates. Many of the code updates are for updating or adding attributes to robots as well as updating the robot battle environment. We consider each collaborator as an independent data node constantly building dyadic relationships with others via code updates to the Robocode main source code base or by sharing code snippets for various robot designs. Consequently, new knowledge is generated when other programmers implement the new updates into their robot functionality or interact with a newly designed robot battleground. We conjecture that the source code network will be impacted by the sharing of code among the collaborators as well as by how efficient the programmers are at searching the source code network for new functionality and successfully interpreting and implementing this new knowledge.

### 4.2. Robocode analysis

When a collaborator submits a new update to a class or package, we can construct a relational matrix between this collaborator and the remaining collaborators in the network by noting if one of the other collaborator reference or use a particular class or package update.<sup>7</sup> Furthermore, this relational matrix can be constructed for each collaborator in the project. The aggregation of the relational matrices can be used to identify sharing of data (classes and packages). If a collaborator references or uses someone’s class or package, this suggests that new information has been generated as a result of a dyadic relations being formed. To get an assessment of a collaborator’s network expansiveness, we can compute a *collaboration index* (CI) Using Equation (1). Additionally, to measure the impact of each collaborator on the overall information content in the network, we can also compute a *collaboration impact value* (CV), Equation (2).

We hypothesize that a collaborator who has many classes or packages that exhibit transitivity will have high impact on knowledge formation. In the Robocode project, we can look at the class(es) or package(s) developed by a single collaborator and count the number of transitive relationships for the

<sup>7</sup> Such a matrix may be rectangular, not square as typically observed in most literature when using a DSM construct.

collaborator. We can also compute a *knowledge influence index (KI)* using Equation (3).

In the case of the Robocode project, we hypothesize that a collaborator with a high *KI*, influences how software modules are built or compiled and aggregated into the final Robocode software application. Consequently, a collaborator with a high *KI* value will likely need to develop their classes and packages with well defined interface boundaries and specification so that other collaborator can integrate their module seamlessly. A drawback to a high *KI* value is the impact of small code changes in a class or package could impact a large number of other collaborators in the development network.

## 5. SUMMARY & CONCLUSION

We propose a framework that illustrates a multi-domain approach to understanding the product development process. This framework suggests that a product can be considered the final outcome when data is successfully transformed into information, information is transformed into knowledge, and this knowledge is applied to creating a product. To facilitate our understanding and modeling of the flows within the various product development networks, we have used network analysis techniques and the DSM. As a case study, we consider the PD process for an open source software development project, called Robocode. This framework attempts to provide additional insights into the argument discussed earlier – do organization design products or do the products design the organization. The best conclusion is to consider that there exists a mutual or two-way exchange in both directions and this framework attempts to separate the various decision stages within this exchange. The framework can help in answering many questions including: What is the important data that the organization must collect? How can we aggregate this data into appropriate information? Are there better arrangements of the data in the databases? Should additional communication links be added in the people network that could enhance the usefulness of the information layer or the application knowledge that could lead to “better” product architectures?

In the current model, we consider only one way flow data → information → knowledge → product although in the “real world” an organization will capture the knowledge gained during each PD project. By implementing knowledge management into the framework can provide a way to “work-out” the impact of design iterations and product architecture changes impact Data→Information→Knowledge→Product overtime. Additionally, we can extend this framework to consider a Data→ Information→ Knowledge→ Product→ Platform. Another possible direction for future work is to capture organizational innovation – where does innovation reside in the PD network, can we modify layers to create additional links or nodes that will increase the innovativeness of an organization.

## REFERENCES

- Abdel-Hamid, T. and S. Madnick., 1991. *Software Project Dynamics, An Integrated Approach*. Englewood Cliffs, NJ. Prentice-Hall, Inc., 1991.
- Ackoff, R. L., 1989. "From Data to Wisdom", *Journal of Applied Systems Analysis*, Volume 16, 1989 p 3-9.
- Aguilar, F. J., 1967. *Scanning the Business Environment*. The Macmillian Company.
- Alexander, C., 1964. *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA, 1964.
- Allen, T., "Architecture and Communication among Product Development Engineers," MIT Sloan School of Management Working paper, No. 3983, 1997.
- Artzy-Randrup, Y., Fleishman, S., Ben-Tal, N., Stone, L., 2004. "Comment on Network Motifs: Simple Building Blocks of Complex Networks and Superfamilies of Evolved and Designed Networks". *Science*, Vol. 5, August 20.
- Batallas, D.A., Yassine, A., 2006. "Information Leaders in Product Development Organizational Networks: Social Network Analysis of the Design Structure Matrix," *IEEE Transactions on Engineering Management*, (43(4) 2006.
- Borgatti, S.P., M.G. Everett, and L.C. Freeman, UCInet for Windows: Software for Social Analysis, Harvard Analytic Technologies, 2002.
- Bradley, J., Yassine, A., 2006. "On the use Of Social Networks for the Analysis of Product Development Teams," ASME Design Theory and Methodology Conference, Philadelphia, Penn. Sept. 10-13.
- Braha, D., 2001. *Data-Mining For Design and Manufacturing*. Kluwer Academic Publishers.
- Braha D., Bar-Yam, Y., "Information flow structure in large-scale product development organizational networks." *Journal of Information Technology* (19) 2004, 244-253.
- Browning, T., Deyst, J., Eppinger, S.D., D.E. Whitney, 2002. "Adding Value in Product Development by Creating Information and Reducing Risk," *IEEE Transaction on Engineering Management*, 49(4) 2002, pp. 443-458.
- Browning, T.R., 2001. "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," *IEEE Transactions on Engineering Management*, Vol. 48(3), 2001.
- Chesbrough, H., Kusunoki, K., 2001. The modularity trap: Innovation, technology phase-shifts, and the resulting limits of virtual organizations." In I. Nonaka and D. Teece (eds.), *Managing Industrial Knowledge: Creation, Transfer and Utilization: 202-230*. Thousand Oaks, CA: Sage.
- Collier, W., 1997. "An integral PIM strategy –Implementation roadmap," DH Brown Associates Inc., PLM Road Map 1997, September 9-11, Dearborn Michigan, 1997.
- Comino, S., Manenti, F., Parisi, M., 2007. "From planning to mature: On the success of open source projects". *Research Policy*, Vol. 36, pp. 1575-1586.
- Danilovic, M., Browning, T., 2007. *Managing complex product development projects with design structure*

- matrices and domain mapping matrices. *International Journal of Project Management*. Vol. 25, pp. 300-314, 2007.
- Eppinger, S., Salminen, V., 2001. "Patterns of product development interactions". *Proceedings of International Conference on Engineering Design, ICED'01*, August 2001.
- Ford, D. and Serman, J. 2003. "Overcoming the 90% Syndrome: Iteration Management in Concurrent Development Projects," *Concurrent Engineering Research and Applications*. Vol. 111, No. 3, pp. 177-186, Sept., 2003.
- Gokpinar, B., Hopp, W., Irvani, S. The impact of product architecture and organization structure on efficiency in quality of complex product development. Unpublished manuscript.
- Gulati, R.K., Eppinger, S.D., 1996. "The coupling of product architecture and organizational structure decisions", working paper.
- Hansen, M.T., 1999. "The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge Across Organization Subunits". *Administrative Science Quarterly*, 44 (1999); pp. 82-111.
- Harary, F. 1969. *Graph Theory*, Reading, MA. Addison-Wesley.
- Hellgren, B., and Stjernberg, T., 1995. "Design and Implementation in Major Investments- A project network approach," *Scandinavian Journal of Management*, 11, 4, 377-394, 1995.
- Henderson, R.M., Clark, K.B., 1990. "Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms". *Administrative Science Quarterly*, 35, 1, pp. 9-30.
- Iyer, B., Gottlieb, R., 2004. "The Four-Domain Architecture: An approach to support enterprise architecture design". *IBM Systems Journal*, Vol. 43, No.3. pp. 587-597.
- King, N. and A. Majchrzak, 1996. "Concurrent Engineering Tools: Are the Human Issues Being Ignored?" *IEEE Transactions on Engineering Management*. 43(2) 189-201, 1996.
- Kogut, B., Metiu, A., 2001. "Open-Source software development and distributed innovation". *Oxford Review of Economic Policy*, Vol. 17. No. 2, pp. 248-264.
- Krackhardt, D., Carley, K., 1998, "A PCANS Model of Structure in Organization" Pp. 113-119 in *Proceedings of the 1998 International Symposium on Command and Control Research and Technology*. Conference held in June. Monterey, CA. Evidence Based Research, Vienna, VA.
- Levitt, R., Thomsen, J., Christiansen, T., Kunz, J., Jin, Y., Nass, C., 1999. "Simulating Project Work Processes and Organizations: Toward a Micro-Contingency Theory of Organizational Design." *Management Science*, Vol. 45, No. 11, pp. 1479-1495, 1999.
- Maurer, M., Lindemann, U., 2007. *Facing Multi-Domain Complexity in Product development. Competence in Design and Development*. January 2007.
- Meier, C., Yassine, A., Browning, T., 2007. "Design Process Sequencing with Competent Genetic Algorithms," *ASME Journal of Mechanical Design*, Vol. 129, No. 6, 2007, pp. 566-585.
- Morelli, M.D., Eppinger, S.D., and Gulati, R.K., 1995. "Predicting Technical Communications in Product Development Organizations", *IEEE Transactions on Engineering Management*. vol. 42, no. 3, pp. 215-222, August 1995.
- Moreno, J.L., 1934. *Who Shall Survive?: Foundations of Sociometry, Group Psychotherapy, and Sociodrama*. Washington, D.C., Nervous and Mental Disease Publishing Co Reprinted in 1953 (Second Edition) and in 1978 (Third Edition) by Beacon House, Inc., Beacon, NY.
- Parashar, S., Bloebaum, C., "Decision Support Tool for Multidisciplinary Design Optimization (MDO) using Mutli-Domain Decomposition," 46<sup>th</sup> AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference, 18-21 April 2005, Austin, Texas.
- Rechtin, E., and Maier, M., 1997. *The Art of Systems Architecting*, CRC Press, Boca Raton, FL, 1997.
- Sanchez, R., Mahoney, J., 1996. "Modularity, Flexibility, and Knowledge Management in Product and Organization Design". *Strategic Management Journal*, (17) December, pp. 63-76.
- Seitamaa-Hakkarainen, P., Lahti, H., Muukkonen, H., Hakkarainen, K., 2000. "Collaborative Designing in a Networks Learning Environment". *Collaborative Design Proceedings of CoDesigning 2000*. Chp 38, pp. 411-420.
- Sharman, D., Yassine, A., Carlile, P., 2002. "Architectural Optimisation using real options theory and dependency structure matrices. ASME 2002 International Design Engineering Technical Conferences. 28<sup>th</sup> Design Automation Conference, Montreal, Canada, September, 29-October 2, 2002.
- Shooter, S., Keirouz, W., Szykman, S., Fenves, S., 2000. "A Model for the flow of design information in product development." *Engineering wth Computers*, Vol. 16, pp. 178-194.
- Simon, H., 1957. "A behavioral model of rational choice" in H.A. Simon (ed.). *Models of Man*. New York: John Wiley, 241-60.
- Simon, H., 1969. *The Sciences of the Artificial*, MIT Press, Cambridge, MA, 1969.
- Sorenson, O, Rivkin, J.W., Fleming, L., 2006. "Complexity of networks and knowledge flow". *Research Policy* Vol. 35, pp. 994-1017.
- Sosa, M.E., 2008. "A structured approach to predicting and managing technical interactions in software development," *Research in Engineering Design*, forthcoming, 2008.
- Sosa, M.E., Eppinger, S.D., Rowles, C.M., 2004. *The Misalignment of Product Architecture and Organizational Structure in Complex Product Development*. *Management Science* (50), No. 12, December, 1674 - 1689.

- Steward, D. 1981. The Design Structure Matrix: A Method for Managing the Design of Complex Systems. EM-28(3) 71-74.
- Ulrich, K., Eppinger, S., 2004. *Product Design and Development*, 3<sup>rd</sup> Edition, McGraw-Hill, Inc., New York, 2004.
- von Hippel, E., 2005. "Democratizing Innovations". The MIT Press, Cambridge, Mass.
- Wasserman, S., Faust, K. 1994. *Social Network Analysis*. Cambridge University Press, New York.
- Wheelwright, S., K. Clark. 1992. *Revolutionizing Product Development*. Free Press: New York, 1992.
- Whitney, D., 2004. "Connectivity Limits of Mechanical Assemblies modeled as Networks". ESD-WP-2004-07
- Yassine, A. A., N. R. Joglekar, D. Braha, S. D. Eppinger, and D. E. Whitney, 2003. "Information Hiding in Product Development: The Design Churn Effect," *Research in Engineering Design*, 14(3), 2003a.
- Yassine, A., Whitney, D., Daleiden, S., Lavine, J., 2003. "Connectivity Maps: Modeling and Analyzing Relationships In Product Development Processes," *Journal of Engineering Design*, 14(3) 2003b.
- Yassine, A., Braha, D., 2003. Complex Concurrent Engineering and the Design Structure Matrix Method. *Concurrent Engineering Research and Applications*, September 2003, Vol. 11, No. 3, pp. 165-176.
- Yu, T., Yassine, A., Goldberg, A., 2007. "An Information Theoretic Method for Developing Modular Architectures Using Genetic Algorithms," *Research in Engineering Design*, Vol. 18(2) 2007, pp. 91-109.