Regular Paper

Evaluating Product Development Systems Using Network Analysis

Shawn T. Collins,[1,*] Ali A. Yassine,[2] and Stephen P. Borgatti[3]

[1]*Control Systems Engineering Department, Rolls-Royce, Indianapolis,* IN 46206

[2]*Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana,* IL 61801

[3]*Department of Management, University of Kentucky, Lexington,* KY 40506

Received 8 January 2008; Accepted 16 April 2008, after one or more revisions
Published online in Wiley InterScience (www.interscience.wiley.com)
DOI 10.1002/sys.20108

ABSTRACT

This paper proposes the integration of two systems engineering analysis tools, the Design Structure Matrix (DSM) and Network Analysis (NA), to study task interactions in a Product Development Process (PDP). The DSM is a matrix-based systems engineering tool that analyzes task sequences to improve PDP execution. Using NA metrics to measure properties of information flow helps to identify important product development tasks and interactions that constrain PDP execution. Project managers can use these data to structure team integration mechanisms or to identify coordinating mechanisms for groups of concurrently scheduled PDP tasks. Functional managers and process architects can use these data to identify important or overloaded tasks. They can also evaluate whether tasks like stage gates and design reviews are acting as effective information flow regulators in the PDP. This new Systems Engineering approach provides a rigorous decision support tool for managers who must alter ideal task sequences due to specific schedule, budget, and expertise constraints encountered on their projects. © 2008 Wiley Periodicals, Inc. Syst Eng

*Author to whom all correspondence should be addressed (E-mail: shawn.collins@alumni.purdue.edu).

This paper is based on material presented in a prior INCOSE conference paper [Collins and Yassine, 2007] and an unpublished PhD dissertation [Collins, 2007]. The authors thank four anonymous reviewers for helpful comments on an earlier draft.

© 2008 Wiley Periodicals, Inc.

Key words: design structure matrix; network analysis; product development execution; task information flow; engineering process improvement

1. OVERVIEW

The product development process (PDP) is a complex system composed of an integrated set of tasks that collectively accomplish a defined objective—e.g., developing a new product [Browning, Fricke, and Negele, 2006; INCOSE, 2004]. As such, many existing systems engineering tools can be used to improve our understanding and analysis of these kinds of systems (i.e., product development processes). Graph based techniques, such as CPM/PERT [Spinner, 1989],[1] IDEF [Kusiak, 1999],[2] and flowcharts, and matrix based techniques, such as DSM and the N^2 method, were successful in capturing the relationships between the tasks in a PDP and in scheduling these tasks accordingly [Browning, 2001; Bustnay and Ben-Asher, 2005]. However, these tools lack the sophistication to uncover the underlying statistical properties of the PDP, such as the average number of links per node (incoming and outgoing), the average path lengths between tasks involved in a cyclic flow of information, and clustering or dispersion metrics reflecting the amount of connectedness between tasks [Braha and Bar-Yam, 2007]. Furthermore, as the number of tasks or ties between tasks increases, the corresponding complexity makes these systems increasingly difficult to analyze.[3] As such, standard System Engineering tools only provide a partial view of the development processes they analyze.

We can enhance our understanding of these systems by using established network analysis (NA) techniques. These methods help uncover previously unnoticed trends and properties in the PDP. For example, it is possible to look at patterns of the overall PDP structure and the location of individual tasks within this structure. This is an important analytical shift from viewing a task as having individually-determined characteristics (as is the case in the above-mentioned Systems Engineering tools) to viewing it as representing emergent properties—emerging from patterns of information flow between tasks.

In particular, using NA techniques to measure properties of information flow (such as degree centrality, influence, and brokerage) provides a methodology to identify important product development tasks and interactions (information flow among two or more tasks) that constrain PDP execution. Project managers can use these data to structure team integration mechanisms [Browning, 1998, 1999] based on the information sharing characteristics of their PDP. They can also use these data to identify coordinating mechanisms for groups of concurrently scheduled PDP tasks. Functional managers and process architects can use these data to identify critical or overloaded tasks based on the input / output information relationships. They can also evaluate whether tasks defined as important in the Systems Engineering literature, such as stage gates, exercise significant control over the PDP.

After an overview of the Design Structure Matrix and Network Analysis in Sections 2 and 3, Sections 4 and 5 use a case study at a small engineering company (Smallcomp) to analyze task interactions in the PDP. This analysis provides insight into critical interactions between key PDP tasks at the overall process, individual design phase, and specific task levels. The results are evaluated visually using standard engineering tools, like control charts, and standard NA tools, like network maps. Each case rapidly condenses the entire PDP to show key individual tasks based on different NA metrics. Section 6 discusses lessons learned, applications, and proposes areas for future work.

2. THE DESIGN STRUCTURE MATRIX

The Design Structure Matrix (DSM) is a system engineering tool that uses matrices to model and analyze complex projects, processes or systems [Browning, 2001; Steward, 1981; Whitney, 1990; Yassine and Braha, 2003]. The collected data in a DSM captures the structure of interactions, interdependencies and interfaces between physical system elements like subsystems and modules, product development teams, and information flow between project tasks. Static DSM analysis uses clustering algorithms to identify groups of coupled components to place in subsystems or frequently interacting individuals to assign to project teams [Sosa, Eppinger, and Rowles, 2004] or product modules [Sharman and Yassine, 2004]. Time-based DSM analysis uses partitioning algorithms that optimally sequence project tasks to minimize the impact of rework and unexpected feedback [Yassine et al., 2003]. The information about task cross-coupling, rework impact, and rework probability identifies a range of possible project durations to assist in planning and resource

[1] Critical Path Method / Program Evaluation & Review Technique.
[2] Integrated DEFinition (IDEF) is a family of methods designed to model the decisions, actions, and activities of a system.
[3] We thank an anonymous *Systems Engineering* reviewer for pointing this out.

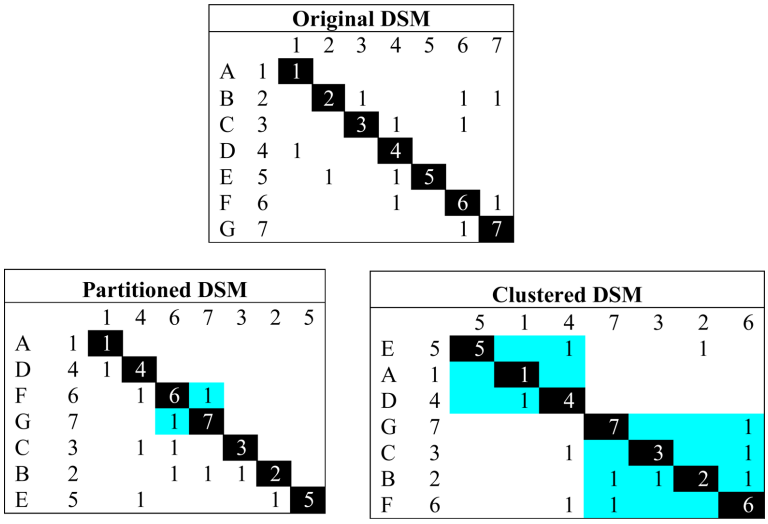

Figure 1. Clustered vs. partitioned DSM analysis. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

allocation. This inclusion of interdependencies and iteration is an improvement over the assumption of linearly independent, sequentially ordered tasks in time based planning tools like MS Project and sequence analysis tools like Value Stream Mapping [Tapping and Shuker, 2003].

Figure 1 shows the two different approaches to analyzing DSM data. The strength of interaction is a two point scale of "Does not interact" (matrix cell is empty) and "Output has some influence" (matrix cell value of 1). The Original DSM has potential for rework because the outputs from tasks G and F provide input to earlier chronological tasks E, C, and B. The Partitioned DSM in Figure 1 has reordered the task execution sequence so that the only feedback loop is between tasks G and F.

One criticism of project management using the DSM is that the task interaction matrix assumes that feedback to earlier tasks is undesirable. This is not true if activities are concurrent or intentionally iterative, as is the case in research and development environments. An alternative approach is to cluster the DSM into task subsets that are mutually exclusive or minimally interacting. Static DSM analysis uses this approach for data that typically do not contain time-dependent information, such as hardware components or team responsibilities. In principal, it can be used to identify groups of R&D tasks that are intentionally iterative or concurrently scheduled. The Clustered DSM in Figure 1 demonstrates the difference in task sequencing compared to the Partitioned DSM. A weakness of this approach for rigorous understanding of information flow in a PDP is that DSM techniques currently do not include methods to evaluate the within and between cluster task relationships once clusters are identified. However, a number of diagnostic metrics are available from Network Analysis (NA).

3. NETWORK ANALYSIS CONCEPTS FOR PRODUCT DEVELOPMENT

Collecting data in network form and applying social network metrics has a number of conceptual benefits for evaluating complex systems [Batallas and Yassine, 2006; Kilduff and Tsai, 2003; Scott, 2000]. First, network analysis focuses on relations and patterns of relations rather than attributes of actors or tasks. Second, it is amenable to multiple levels of analysis, allowing insight into micro-macro linkages. For example, while an overall network influences patterns of individual members, individual actions create subgroups that exert influence for a particular type of relationship. The network approach thus provides insight into how individual elements affect institutions they are part of, and how institutions constrain their individual elements.

One major benefit is that the overall structure tends to be emergent, in that no individual in the network understands the entire system. Network analysis uses patterns of ties to define and analyze the emergent structure. This is a significant benefit when there is no "right answer" about what the structure should look like. Finally, network analysis integrates quantitative, qualitative, and graphical data, thus allowing more thorough and in depth analysis.

The basic element of NA is an adjacency matrix that defines relationships between each row and column

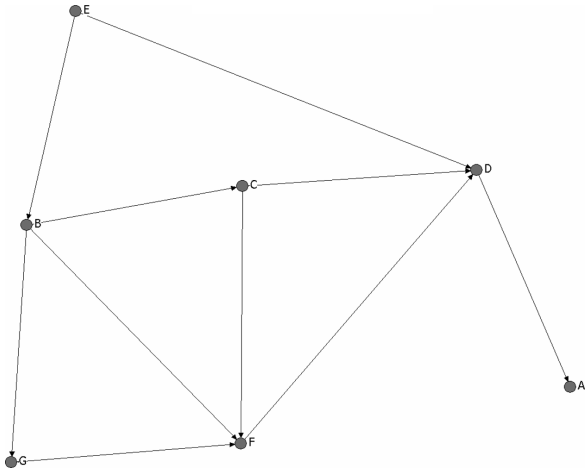

Figure 2. Network map of Figure 1.

element. A one-mode matrix is built when the row and column elements are the same, as with frequency of interaction among a group of people or distance between cities. For symmetric one-mode data, the upper and lower halves of the matrix are identical. For directed one-mode data, the two halves are different. Directed data is collected on ties that are not reciprocal, as with authority structures or transfer of advice. For a matrix of product development tasks collected in DSM format, directed data indicates the input / output flow of information between each task. Figure 2 shows the network drawn from the data in Figure 1. The arrows indicate a tie from the row task to the column task. For example, task A receives output from task D, while task B provides output to tasks C, F, and G.

Once data are collected in matrix form, NA methods contain a variety of analytic techniques [Chartrand, 1985; Hanneman and Riddle, 2005; Scott, 2000; Wasserman and Faust, 1999] that have been used to study topics as diverse as world systems [Brieger and Pattison, 1981; Snyder and Kick, 1979], diffusion of innovation [Burt, 1978, 1987], production of scientific knowledge [Brymn, 1988], and organizational influence [Galaskiewicz and Krohn, 1984; Roethlisberger and Dickson, 1939]. These methods target three different levels of analysis [Kilduff and Tsai, 2003] First, they examine patterns of the overall structure and the location of individuals within the structure. Second, they identify substructures (groups of nodes that are closer to each other than to other groups) based on patterns of relations. Examples are blocks, factions, cliques, and bridges. Third, they evaluate positions of individuals within the structure. This is an analytical shift from viewing a node as having individually unique characteristics to viewing it as representing some categorical attribute. Demographic attributes are based on variable similarity like age (for people) or time to complete (for tasks). Network position analysis looks at attributes emerging from patterns of ties between nodes. Network categories are thus inherently relational roles rather than individual categories. Table I summarizes a set of key NA metrics for evaluating interacting product development tasks [Wasserman and Faust, 1999]. The remaining sections of this paper discuss these metrics in more detail.

Table I. Network Analysis Metrics for Evaluating PDP Task Interactions

NA Term	Definition	Product Development Application
Analysis Level: Overall Structure		
InDegree Centrality	Number of incoming ties for each node	Critical PDP task based on high information *collection* load
OutDegree Centrality	Number of outgoing ties for each node	Critical PDP task based on high information *dissemination* load
Analysis Level: Sub-Structure		
Clusters	Divide the network into N groups with maximized internal ties and minized external ties	Identifies the N groups of iterative tasks that can be conducted concurrently
Density	Number of actual ties between nodes or groups divided by number of possible ties	Measures iteration within a group of concurrently scheduled tasks and between groups of separate tasks
Analysis Level: Individual Tasks		
Influence	Degree of network dependence on a node (teacher of the next president)	Low visibility PDP tasks that have potentially important information
Betweenness Centrality	Number of times a node is on the shortest path between two other nodes	PDP "transistor" task that regulates information flow between two other tasks
Brokerage	What is the nature of B's role as a transmitter of information between A and C (co-ordinator, gatekeeper, representative, liason, consultant)?	Characterizes role of key tasks that transfer information within and between PDP task groups



Figure 3. Standard stage-gate system.

4. NETWORK ANALYSIS OF PRODUCT DEVELOPMENT TASKS

This section discusses network analysis of Product Development tasks interactions at a small engineering company (Smallcomp). Smallcomp uses a common stage gate system [Cooper, Edgett, and Kleinschmidt, 2002; Cooper and Kleinschmidt, 1993] in Figure 3 for its product development activity. The business gate reviews are held after each design phase to ensure that the maturing technical design still matches the expected market characteristics anticipated for the product deployment. This is initially done at Business Gate 0 to identify a market opportunity. It is done at Business Gate 4 to formally release the product for sales. Each business gate is preceded by an engineering design review to evaluate the product's technical feasibility and risk relative to the defined requirements. The Concept Review is held before Business Gate 1, Preliminary Design Reviews are held before Business Gate 2, the Critical Design Review is held before Business Gate 3, and the Validation Review is held before Business Gate 4.

Smallcomp began investigating matrix based project management tools during a quality improvement initiative to streamline its product development task execution. This resulted in their PDP being captured and evaluated using a traditional time based DSM, as shown in Figures 4 and 5.[4] The remainder of this section uses those DSM data to discuss how the NA diagnostics in Table I can provide insight that guides product development activity.

4.1. Overall PDP Structure

At this level of analysis, NA metrics treat the product development process as a single entity of tasks that share information. Table I identifies two key metrics. InDegree Centrality measures the number of inputs that

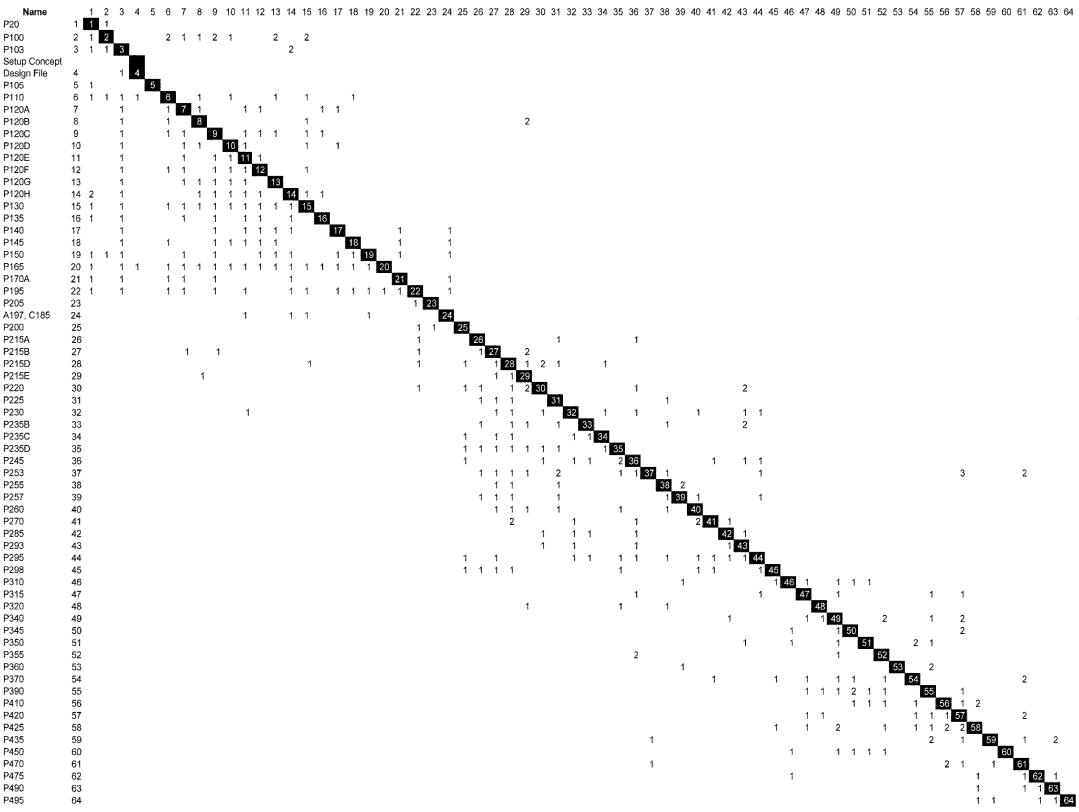

Figure 4. Full DSM of Smallcomp's PDP.

[4] 1 denotes a weak task dependency. 2 denotes a strong dependency. The lower half of the matrix shows feedforward relationships. The upper half shows feedback relationships. We use this designation to maintain consistency with NA convention.

Name		1	2	3	4	5	6	7	8	9	10	11	12
P20: Business Gate 0: Approval to develop concept design	1	1	1										
P100: Customer Specification Review: Reconciled customer requirements, design requirements	2	1	2				2	1	1	2	1		
P103: Establish Organization for Conceptual Design: Assign: Program Manager Project Manager Required PPI/IPD Team Leads / Members	3	1	1	3									
Setup Concept Design File: Concept Design Approval List, Charge Numbers	4			1	4								
P105: Review Lessons Learned Database: Power plant level prelim. design LL checklist	5	1				5							
P110: Generate Power Plant Design Requirements: PPR's, High Level Power Plant Objectives (Preliminary PPR)	6	1	1	1	1		6		1		1		
P120A: Define Process (flow) System: Flow schematic	7			1			1	7	1			1	1
P120B: Define Design Points: Design conditions	8			1			1		8				
P120C: Define Electrical System: Electrical block diagram	9			1			1	1		9		1	1
P120D: Predict Design Condition Performance: Basic system heat & mass balances	10			1				1	1		10	1	
P120E: Identify Major Assemblies: Identification of major assemblies	11			1				1		1	1	11	1
P120F: Define Power Plant Package: Drawing showing component arrangement	12			1			1	1		1	1	1	12

Figure 5. DSM excerpt for Smallcomp's concept phase without optimal sequencing.

each task has. This defines the information collection load in the PDP. OutDegree Centrality measures the number of outputs that each task has. This defines the information dissemination load in the PDP. It is one metric of task criticality or workload. Figure 6 shows control charts for Degree Centrality based on the number of incoming ties (InDegree) and outgoing ties (OutDegree). The axis progression is from beginning to end of the project execution. The upper and lower control limits (UCL, LCL) are drawn based on the two-sigma point around the average value. Project managers reviewing these data would need to plan that each task to be completed generally requires input from between four and ten other tasks, and provides output to between two and eleven other tasks. This can help them plan the integration mechanisms for the individuals or teams who must collect information to conduct their tasks, or provide results of their completed tasks to others.

Figure 6 rapidly summarizes the outlier tasks that have relatively high or low information sharing loads. Relative to the other tasks in the PDP, tasks above the Upper Control Limit have a higher than average role to coordinate either incoming or outgoing information. The tasks with high InDegree Centrality, meaning they require more than the normal information collection burden, are the customer specification review (P100), design and development plan (P150), concept design review (P165), Business Gate 1 (P195), and test stand preparation (P253). In Smallcomp's case, assigning above average information collection load is correct in the sense that effective execution requires collecting inputs from many other tasks. The exception is task P253. Smallcomp's process architects compensated for historical rework associated with test stand design by requiring task P253 to collect input from a large number of other tasks. However, discussions with Smallcomp's engineers revealed that the large information collection burden signified by placing task P253 above the control limits had not reduced the churn associated with test stand design.

The tasks with low InDegree Centrality, meaning they require less than the normal input, are Business Gate 0 (P20), reviewing lessons learned (P105), establishing design approvals and charge numbers for preliminary design (P205). P20 and P205 are legitimately outside the control limits that effective execution requires collecting inputs from relatively few other tasks. The low value for P105 shows the unclear interaction of knowledge management with other tasks when Smallcomp defined its PDP. However, P205, which is key to the financial health of the program, is also below the control limits. Thus, low InDegree Centrality does not necessarily mean the task is not important. This is an important insight, as process improvement activities frequently focus on removing task links (shift the out-

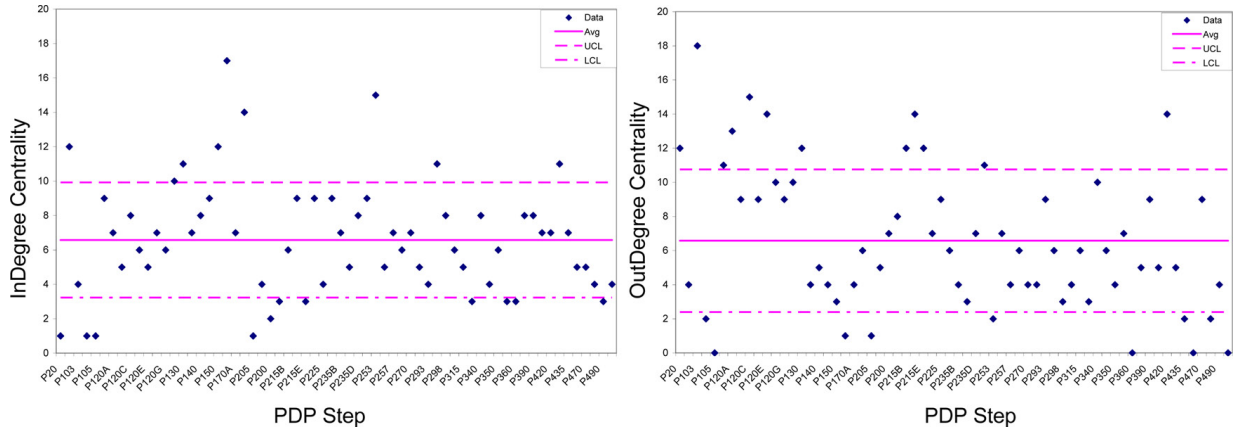

Figure 6. Control chart for degree centrality. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

liers within the control limits), or else eliminating tasks with few links (remove the low outliers).

The control chart in Figure 6 quickly identifies two improvement opportunities by examining tasks outside the information collection and dissemination control limits. The first questioning whether the most effective way to address task P253's importance is to give it a relatively high information collection burden. The second is identifying the need to increase ties from P105 so that lessons learned reviews are better integrated with the other design activities.

4.2. PDP Substructures

At this level of analysis, NA metrics identify and measure groups of tasks that interact more with each other than with other tasks in the PDP. Table I identifies two key metrics. Clustering identifies groups of tasks that are strongly connected to each other as a basis for scheduling concurrent activity. Density measures the number of actual and potential ties within a cluster, and between clusters. This describes the strength of concurrency, as well as the strength of coupling between groups of tasks.

Table II divides Smallcomp's PDP into five task clusters.[5] The algorithm maximizes within-cluster task

Table II. Clusters of Concurrent PDP Tasks

	System Definition	Preliminary Functional Integration	Preliminary Mechanical Integration	Detailed Design & Verification	Validation
System Definition	**0.41**	0.03	0.02	0.00	0.06
Preliminary Functional Integration	0.01	**0.39**	0.18	0.05	0.03
Preliminary Mechanical Integration	0.00	0.11	**0.37**	0.05	0.00
Detailed Design & Verification	0.00	0.02	0.00	**0.45**	0.10
Validation	0.08	0.02	0.00	0.03	**0.16**

[5]The upper half of the table shows feedforward ties. The lower half shows feedback ties.

interaction and minimizes between-cluster task interaction. The diagonal values in Table II measure the density of ties within the cluster. The off-diagonal measures the density of ties between clusters. For example, there are 41% of the potential connections within the System Definition phase, 3% of the potential connections from System Definition outputs to Preliminary Functional Integration inputs, and 1% of potential connections from Preliminary Functional Integration outputs to System Definition inputs. Several tasks conducted early in PDP execution (P20—Business Gate 0, P100—Customer Specification Review, Set up Design File, and P105—Lessons Learned) are placed in the Validation cluster based on loose ties to the rest of the PDP network. This placement shows that the relationship of these tasks to other System Definition tasks was not well defined in Smallcomp's original DSM.

The low off-diagonal density values in Table II show that the clustering intent, which was to identify groups of iterative or concurrent tasks that are independent of the rest of the PDP activities, was mostly successful. The primary exception is the ties between the Preliminary Functional and Mechanical Integration phase. The Functional–Mechanical tie is relatively strong (18% between-phase compared to 39% and 37% within-phase), while the Mechanical–Functional tie is slightly weaker (11% between-phase compared to 39% and 37% within-phase). Both are considerably higher than the between-phase ties indicated by the remainder of the off-diagonal values in Table II. Project planning would define a single set of activities called Preliminary Integration, which would need coordinating mechanisms in place to manage the interdependencies between these two sets of concurrent activities (Functional and Mechanical Integration). The interdependencies are defined by determining which tasks links create the off-diagonal density values in Table II.

4.3. Individual PDP Tasks

At this level of analysis, NA metrics identify key tasks based on their relational roles as information transmitters in the product development process network. Table I identifies four metrics. Betweenness centrality measures the number of times a particular task is on the shortest path between two other tasks. The steps with the highest betweenness centrality are Design Point Definition (P120B), Detailed Heat and Mass Balance (P215E), P&ID and Electrical 1-line (P215B), Business Gate 1 (P195), and Audit Preliminary Design (P215D). Like standard critical path identification, tasks with high betweenness centrality have a high schedule impact on the overall PDP if they are not completed on time.

As a measure of shortest information flow, betweenness centrality also identifies which tasks have the greatest control of information in the PDP. Measuring betweenness centrality provides a way to determine whether tasks that are expected to be critical for PDP execution actually control information flow in the PDP task input / output relationships. The tasks identified here show that Smallcomp has a strong focus on tasks that control the transition between from the Concept to Preliminary Design phases. Tasks performed later in the PDP, such as at the transition from Detailed Design to Verification, do not score high on this measure. Business Gates 2 and 3, which Figure 3 defines as critical to managing PDP execution, also do not score high on betweenness centrality.

Centrality evaluates the number of immediate ties that each PDP task has. Influence measures the degree of connectedness between each task and every other task in the PDP. Tasks with high influence have information that is important to the entire PDP, even if they do not appear on the critical path. Tasks with low influence have information that is not important to the entire PDP. The most influential steps are P215E (Detailed Heat and Mass Balance) and P120B (Design Point Definition). It makes intuitive sense that the steady state model and design points would be highly influential, because these concept level activities contain information that constrains shapes execution for the rest of the PDP. The least influential steps are P120F (Define Power Plant Package) and P120E (Identify Major Assemblies). This shows the low priority given to physical packaging during concept definition. The low priority stems from Smallcomp's heritage of creating demonstration units where physical space was not typically a design concern.

Another metric of individual position is *brokerage*. Brokerage occurs for the triad of nodes A, B and C in Figure 7. In this network, B is a broker because it passes information from A to C. Node B has three potential brokerage roles between nodes A and C. B is a *coordinator* if all three nodes are in the same group. B is a *gatekeeper* if A is in a different group from B and C. B is a *representative* if C is in a different group from A and B.

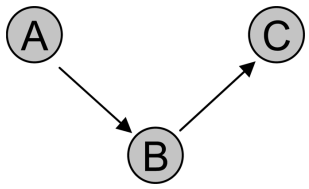

Figure 7. Example brokerage by node B.

Brokerage identifies tasks that have critical information sharing roles within and between the task clusters in Table II. Expected brokerage calculates the number of times each task in a cluster would be expected to play each brokerage role based on the number of clusters and size of each cluster. Relative brokerage ratios the actual number of roles each task plays against the expected number of roles. High relative brokerage values indicate a task is playing a particular role more than expected. Low relative brokerage indicates the task is playing a role less then expected.

Figure 8 shows the number of tasks in each PDP cluster where the relative brokerage value is greater than 3. The type of coordination varies across different clusters of PDP tasks. For example, the high coordination roles in all five task clusters identify tasks that manage design iteration in each phase. This attention to the required coordination *within* a PDP phase is expected based on Smallcomp's desire to define clusters of concurrently scheduled or iterative activities. The coordination burden decreases from System Definition to Functional Integration, with an increase during Detailed Design and decrease during Validation. This shows a cyclical pattern of formal iteration as the design progresses through the PDP.

The other two brokerage roles identify tasks that are critical for information transfer between PDP phases. The increasing Representative values from System Definition through Mechanical Integration shows a larger number of tasks involved in passing information to another PDP phase. The increasing Gatekeeper values show a larger number of tasks involved in receiving information from another PDP phase. Following Functional Integration, the Representative role reduces as there are fewer phases receiving design information. However, the Gatekeeper role increases for the Validation, showing the importance of receiving new information.

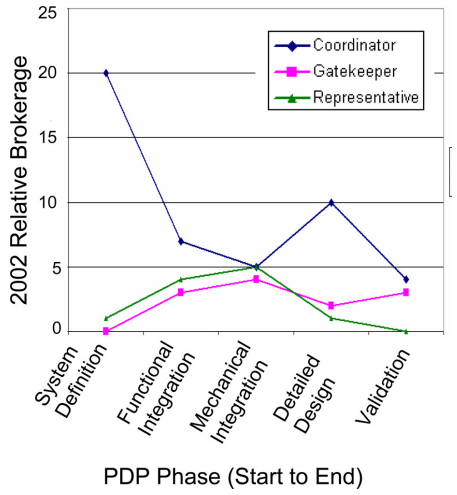

Figure 8. Relative brokerage rules in Smallcomp's PDP phases. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Many programs will modify the baseline PDP based on external constraints such as compressed schedule, reduced resources, or altered scope. This increases the burden of coordinating information transfer between program participants. Brokerage roles are a standard set of NA metrics that can be used to facilitate discussions about the risk management, design iteration, and configuration control required to coordinate information transfer between these teams. This is particularly valuable when tasks are being completed by distributed teams.

5. VISUALIZING INFORMATION FLOW PATTERNS IN THE PDP

A final benefit of NA techniques is the ability to visually display large systems of interaction captured in matrix form. In this section, we use the visualization tools to identify insights from displaying PDP tasks sequentially (based on expected flow of project execution), and based on key NA metrics discussed in Section 4.

Figure 9 shows a network map of Smallcomp's entire PDP based on the DSM data in Figure 4. Each node is an individual task in the PDP. Lines between nodes indicate an information flow dependency between tasks. The arrow shows the direction of information flow. The node shape is defined in Table III. The task execution in Figure 9 moves from right (early program activity) to left (field testing and support). However, the order is clearly not a linear sequence. Traditional process management techniques such as value stream mapping [Tapping and Shuker, 2003] which rely on visual process description to identify key process characteristics are difficult to apply in this case due to the dense interconnections between tasks.

Graph theory methods for visualizing networks are a powerful way to display information within a PDP. For example, overlaying the business gate process with the task interactions in Figure 9 clearly shows that the business gates are not acting as the checkpoints they are supposed to. One external reviewer of an earlier version of this paper [Collins and Yassine, 2007], on seeing this figure, commented: "You probably do not have a way to control projects that are in between Gate 1 and Gate 4, and you probably have significant cyclical activity between your design, verification, and validation phases." During a process reengineering activity conducted with a group of Smallcomp managers, several commented: "We don't really understand the purpose of Business Gates 2 and 3." The confirmation of com-

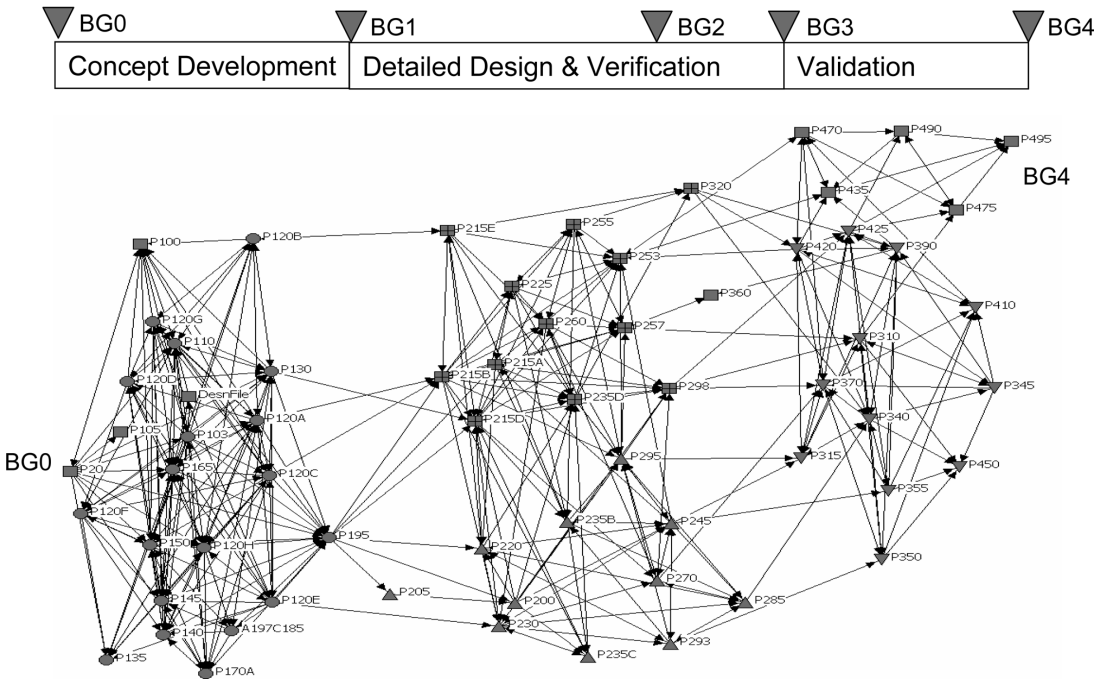

Figure 9. Network map of Smallcomp's PDP.

ments from Smallcomp's managers by a reviewer who, despite no knowledge of Smallcomp's specific processes or products, was able to make a remarkably accurate assessment of Smallcomp's recent history, suggests the underlying value of NA visualization methods. This history would not be immediately obvious from the business gate flow chart in Figure 3, nor would it be clear from the classical DSM matrix display in Figure 4.

The visual display in Figure 9 confirms the finding from the centrality control chart in Section 0, which showed that the tasks for Business Gates 2 and 3 were not acting as the check points they were intended to be. There are two options for PDP management. First, redesign the PDP to increase the information control authority of these tasks. Second, use the brokerage analysis from Figure 8 to look at the tasks that are acting as gatekeepers between PDP phases, and focus energy on making sure those steps are robust.

Figure 10 shows a redrawn PDP network map from Figure 9. The *x*-axis places tasks by increasing influence. The *y*-axis places tasks by increasing betweenness centrality. It visually summarizes important tasks based on two NA metrics that are important to PDP execution. Figure 10 condenses interactions between all the PDP tasks to show the six or seven critical tasks. The uniqueness of these tasks would not be immediately obvious using a swimlane, value stream map, or even normal network map. These tasks would be hard to identify in Figure 9.

Filtering a small list of tasks from the entire PDP enables more effective discussions about improving PDP execution. Are the highlighted tasks as important as the filter claims? If so, do they get appropriate attention during program execution? If not, what changes need to be made on future programs? Where are the steps we claim are critical, like technical and business reviews? Should they be showing up with high values on key analytical dimensions?

For the relationships in Figure 10, there would be two questions. First, what tasks should have high con-

Table III. Task Node Shapes in Figures 9 and 10

PDP Cluster in Table 2	Node Shape
System Definition	Circle ○
Preliminary Functional Integration	Triangle △
Preliminary Mechanical Integration	Crossed Square ⊞
Detailed Design & Verification	Inverse Triangle ▽
Validation	Square □

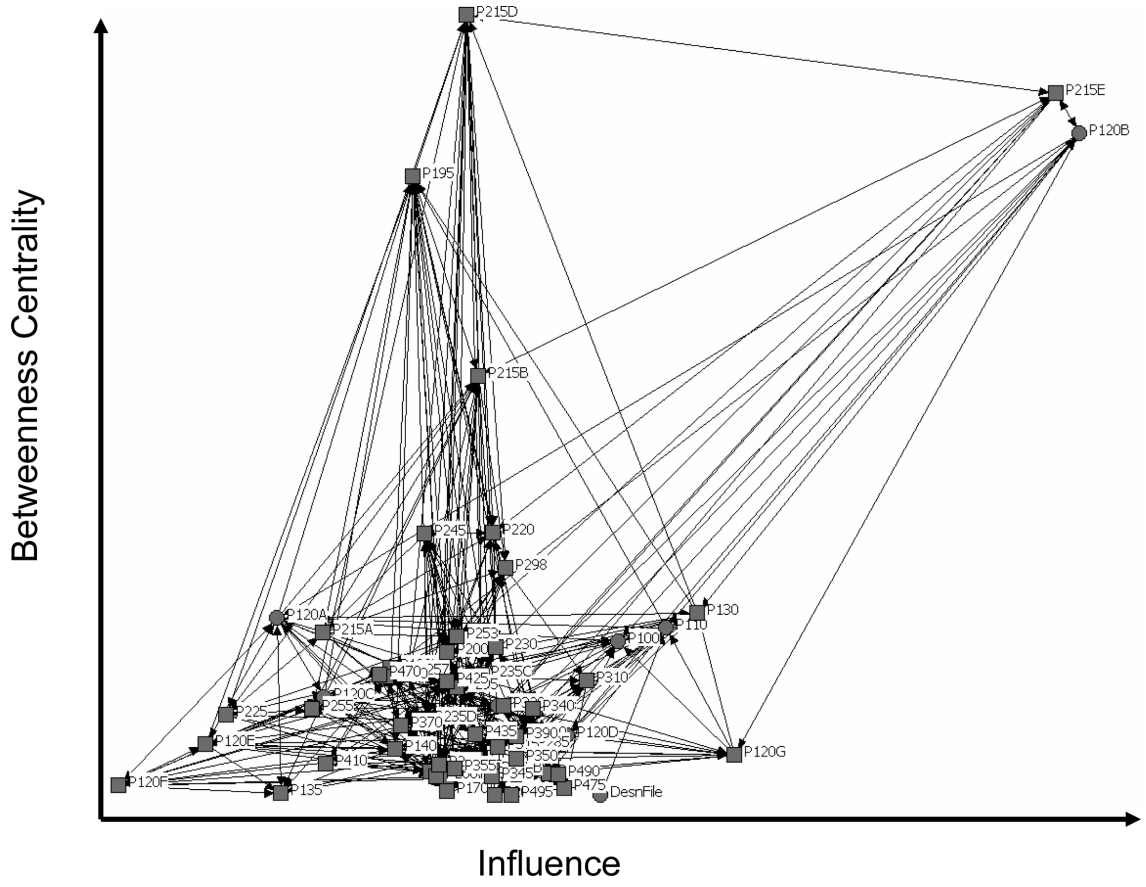

Figure 10. Displaying PDP task relationship along key dimensions.

trol over information flow through the PDP (betweenness centrality)? Second, what tasks should have the ability to reach the entire PDP with important design information (influence)? Figure 10 was a valuable starting point for discussions with a group of Smallcomp's Engineering managers. It validated earlier experience about costly rework resulting from poor design point definition. The discussion also highlighted the lack of formal attention previously paid to assembly drawings. Attributing low influence to P120E and P120F was consistent with legacy programs where packaging was not a major design constraint. They suggested the two steps be reexamined since packaging had become a major design constraint for Smallcomp's programs.

6. DISCUSSION AND CONCLUSION

6.1. Does PDP Documentation Accurately Describe PDP Execution?

Smallcomp's business gate system was defined by a group of experienced engineers responsible for identifying a development process that would represent industry best practices. When the original DSM work was done, the same experts were interviewed to identify individual task interaction patterns. This bottoms-up process definition did not immediately result in the well-coordinated interaction patterns expected from the top down definition. These discrepant process definitions highlight the difficulty of getting the same answer when you ask someone how they think a PDP *should* be executed (Fig. 3), and when you ask someone what they think *actually* happens (Fig. 9). While this type of discrepancy is well documented [Hutchins, 1995], the danger in product development contexts is that some rework gets created because of how individuals executing tasks are asked to share information. The analysis in this paper shows the importance of Systems Engineering tools to rigorously evaluate these complex tasks interactions.

6.2. Future Work

One fundamental tenet of Quality Management [Deming, 1986; Devor, Chang, and Sutherland, 1992; Shewhart, 1931; Taguchi and Wu, 1980; Taylor, 1911] is the importance of data-driven decisions for process

control. With the shift in emphasis upstream of manufacturing to product design [Nichols, 1992], a growing body of literature emphasizes the importance of improving processes that involves creation and transfer of design knowledge [Cooper, 1994; Cooper, Edgett, and Kleinschmidt, 2002; Cooper and Kleinschmidt, 1993; Tapping and Shuker, 2003; Thomke and Reinersten, 1998]. In this context, transmitting information between groups is a significant element of engineering program execution. Managers seeking to make meaningful improvements in the product development cycle may often incorrectly identify actual sources of interdependencies because of the complex and interacting nature of organizational processes, tasks, and relationships [Brehmer, 1992; Repenning, Conclaves, and Black, 2001; Siggelkow, 2002; Sterman, 1989]. It is a significant challenge to understand the underlying patterns in the complex interactions shown by a full DSM (Fig. 4) or network map (Fig. 9).

Network Analysis offers a suite of metrics to analytically measure the information flow characteristics in complex product development environments, rather than relying on an individual or set of individuals to accurately understand the entire process. This allows the PDP to be emergent, rather than predefined phenomena. In particular, the NA metrics in this paper identify important tasks and task interactions in a PDP. Project managers can use these data to structure team integration mechanisms or to identify coordinating mechanisms for groups of concurrently scheduled tasks. Functional managers and process architects can use these data to understand which tasks and interactions truly constrain information flow in the PDP. They can then evaluate whether tasks like stage gates and design reviews are acting as effective information flow regulators in the PDP.

Real time adjustment to ideal sequences is often a fact of life due to schedule, budget, and expertise constraints. We believe that using NA metrics to evaluate information flow provides tools that help managers responsible for making decisions in these contexts. Indeed, metrics that we have not discussed in this paper, such as equivalence [Wasserman and Faust, 1999], reachability [Wasserman and Faust, 1999], and key players [Borgatti, 2006], provide additional opportunities to develop robust measures of information flow in complex product development environments.

REFERENCES

D. Batallas and A. Yassine, Information leaders in product development organizational networks: Social network analysis of the design structure matrix, IEEE Trans Eng Management 43(4) (2006), 570–582.

S.P. Borgatti, Identifying sets of key players in a social network, Comput Math Org Theory 12(1) (2006), 21–34.

D. Braha and Y. Bar-Yam, The statistical mechanics of complex product development: Empirical and analytical results, Management Sci 53(7) (2007), 1127–1145.

B. Brehmer, Dynamic decision making: Human control of complex systems, Acta Psychol 81(3) (1992), 211–241.

R.L. Brieger and P.E. Pattison, "Structures of economic interdependence among nations," Continuities in structural inquiry, P.M. Blau and R. Merton (Editors), The Free Press, New York, 1981, pp. 353–380.

T.R. Browning, Integrative mechanisms for multiteam integration: Findings from five case studies, Syst Eng 1(2) (1998), 95–112.

T.R. Browning, Designing system development projects for organizational integration, Syst Eng 2(4) (1999), 217–225.

T. Browning, F. Fricke, and H. Negele, Key concepts in modeling product development processes, Syst Eng 9(2) (2006), 104–128.

T.R. Browning, Applying the design structure matrix to system decomposition and integration problems: A review and new directions, IEEE Trans Eng Management 48(3) (2001), 292–306.

R.J. Brymn, "Structural location and ideological divergence: Jewish Marxist intellectuals in turn-of-the-century Russia," Social structures, B. Wellman and S.D. Berkowitz (Editors), Cambridge University Press, Cambridge, 1988, pp. 359–379.

R.S. Burt, Cohesion versus structural equivalence as a basis for network subgroups, Sociol Methods Res 7 (1978), 189–212.

R.S. Burt, Social contagion and innovation: Cohesion versus structural equivalence, Amer J Sociol 92 (1987), 1287–1335.

T. Bustnay and J. Ben-Asher, How many systems are there?—using the N^2 method for systems partitioning, Syst Eng 8(2) (2005), 109–118.

G. Chartrand, Introductory graph theory, Dover, Mineola, NY, 1985.

S.T. Collins, Expertise, disagreement, and bad things: Turbulent knowledge amidst organizational stress in engineering R&D, Unpublished Ph.D. Dissertation, University of Connecticut, Storrs, 2007.

S.T. Collins and A.A. Yassine, Evaluating product development task interactions using network analysis, Proceedings of the Seventeeth Annu Int Symp Int Council Syst Eng (San Diego, CA), Seattle: INCOSE, June 24–27, 2007.

R.G. Cooper, Perspective: Third-generation new product processes, J Prod Innovation Management 11(1) (1994), 3–14.

R.G. Cooper, S.J. Edgett and E.J. Kleinschmidt, Optimizing the stage-gate process: What best practice companies do—II, Res Technol Management 45(5) (2002), 21–27.

R.G. Cooper and E.J. Kleinschmidt, Stage gate systems for new product success, Market Management 1(4) (1993), 10–29.

W.E. Deming, Out of the crisis, MIT Center for Advanced Engineering Studies, Cambridge, MA, 1986.

R.E. Devor, T. Chang, and J.W. Sutherland, Statistical quality design and control, Prentice Hall, Upper Saddle River, NJ, 1992.

J. Galaskiewicz and K.R. Krohn, Positions, roles, and dependencies in a community integration system, Sociol Quart 25 (1984), 527–550.

R.A. Hanneman and M. Riddle, Introduction to social network methods, University of California, Riverside, 2005, http://faculty.ucr.edu/~hanneman/).

E. Hutchins, Cognition in the wild, The MIT Press, Cambridge, MA, 1995.

INCOSE, Systems engineering handbook: A "what to" guide for all SE practitioners, International Council on Systems Engineering, Seattle, 2004.

M. Kilduff and W. Tsai, Social networks and organizations, Sage, Thousand Oaks, CA, 2003.

A. Kusiak, Engineering design: Products, processes, and systems, Academic, San Diego, CA, 1999.

K. Nichols, Better, cheaper, faster products - by design, J Eng Des 3(3) (1992), 217–228.

N.P. Repenning, P. Goncalves and L.J. Black, Past the tipping point: The persistence of fire-fighting in product development, Calif Management Rev 43(4) (2001), 44–63.

F.J. Roethlisberger and W.J. Dickson, Management and the worker, Harvard University Press, Cambridge, MA, 1939.

J. Scott, Social network analysis: A handbook, Sage, London, 2000.

D. Sharman and A. Yassine, Characterizing complex product architectures, Syst Eng 7(1) (2004), 35–60.

W. A. Shewhart, Economic control of quality of manufactured product, Van Nostrand, Princeton, NJ, 1931.

N. Siggelkow, Misperceiving interactions among complements and substitutes: Organizational consequences, Management Sci 48(7) (2002), 900–917.

D. Snyder and E.L. Kick, Structural position in the world system and economic growth, 1955–1970: A multiple-network analysis of transnational interactions, Amer J Sociol 84 (1979), 1096–1126.

M.E. Sosa, S.D. Eppinger and C.M. Rowles, The misalignment of product architecture and organizational structure in complex product development, Management Sci 50(12) (2004), 1674–1689.

M. Spinner, Improving project management skills and techniques, Prentice-Hall, Englewood Cliffs, NJ, 1989.

J.D. Sterman, Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment, Management Sci 35(3) (1989), 321–339.

D.V. Steward, The design structure system—a method for managing the design of complex systems, IEEE Trans Eng Management EM-28 (August 1981), 71–74.

G. Taguchi and Y. Wu, Introduction to off-line quality control, Central Japan Quality Control Association, Nogoya, Japan, 1980.

D. Tapping and T. Shuker, Value stream management for the lean office, Productivity Press, New York, 2003.

F. Taylor, The principles of scientific management, Harper, New York, 1911.

S. Thomke and D. Reinersten, Agile product development: Managing development flexibility in uncertain environments, Calif Management Rev 41(1) (1998), 8–30.

S. Wasserman and K. Faust, Social network analysis methods and applications, Cambridge University Press, Cambridge, 1999.

D. Whitney, Designing the design process, Res Eng Des 2(1) (1990), 3–13.

A. Yassine and D. Braha, Four complex problems in concurrent engineering and the design structure matrix method, Concurrent Eng Res Appl 11(3) (2003).

A. Yassine, N. Joglekar, D. Braha, S. Eppinger, and D. Whitney, Information hiding in product development: The design churn effect, Res Eng Des 14(3) (2003), 145–161.



Shawn T. Collins holds the position of Specialist, Controls Systems in the Engineering department at Rolls-Royce in Indianapolis. His professional and academic work focuses on applying anthropological insight to engineering product development. This has led to anthropology interests in ethnographic methods, social network analysis, and cognitive anthropology; and to engineering interests in product development methods, process control, systems engineering, and team performance. Dr. Collins received the B.S. degree in Mechanical Engineering from Purdue University in 1999, the M.S. degree in Mechanical Engineering in 2004 from Renssalaer Polytechnic Institute, and the M.A. and Ph.D. degrees in Anthropology in 2003 and 2007 from the University of Connecticut. He is a member of INCOSE, the Society for Applied Anthropology and the American Anthropological Association.



Ali A. Yassine is an assistant professor in the Department of Industrial and Enterprise Systems Engineering (IESE) at the University of Illinois at Urbana-Champaign (UIUC) and the director of the Product Development Research Laboratory. His research involves managing the development process of complex engineering systems, design process modeling, and IT-enabled concurrent engineering. Dr. Yassine received the B.E. degree in Mechanical Engineering in 1988 from the American University of Beirut and the M.S. and Ph.D. degrees in Industrial and Manufacturing Engineering in 1989 and 1994 from Wayne State University in Detroit, MI. He is a member of INFORMS, ASME, and PDMA.



Stephen P. Borgatti is a professor at the University of Kentucky in the Gatton College of Business and Economics, where he holds the Chellgren Endowed Chair in the Management department. His research interests and consulting activities include knowledge management, culture and cognition and mathematical/computational approaches to organizational theory. He is author of the most widely used social network analysis software, UCINET, and is a 2-term past president of INSNA, the professional association for social network researchers. Dr. Borgatti received the B.A. degree in Anthropology in 1977 from Cornell University and the Ph.D. degree in Mathematical Social Science in 1988 from the University of California, Irvine. He is currently a Senior Editor of Organization Science, and sits on the editorial boards of several social science journals.