A Knowledge-Driven, Network-Based Computational Framework for Product Development Systems

Today’s fast-paced product development (PD) environment brings many new challenges to the PD community. These challenges are mainly due to a drastic increase in the scale and complexity of engineered systems, which require the collaboration of functionally and geographically distributed resources within and outside a firm’s boundary. To address these new challenges, this paper proposes a novel theoretical and computational framework for an enterprise-wide PD management system. The proposed framework considers an integrative view of the various dependencies that co-exist in three PD domains (i.e., people, products, and processes). Additionally, it provides a computational tool that links them together in a succinct and tractable way and provides an analysis method for assessing their influence on shaping the product development process. Using this framework, we suggest that the characteristics of how an organization acquire data, interpret information, and apply knowledge will impact the final architecture of a product. We demonstrate this framework by analyzing the development efforts for a software project called ROBOCODE. [DOI: 10.1115/1.4023166]

Keywords: product development, design decomposition, product architecture, information integration, network analysis

1 Introduction

Developing complex engineered systems typically requires multidisciplinary, interdependent elements that cannot be embraced by a single person, group, or organization. Decomposition, dispersion, and collaboration epitomize a PD endeavor. Historically, the complexity of engineered systems was gauged solely on technical aspects, where the use of system/product decomposition principles to manage such complexity has served the design community well thus far [1–3]. However, today’s PD environment is an amalgam of product/people/processes and the flows of data, information, and knowledge between them resulting in a complex system of interactions as depicted in Fig. 1. This intricate system presents new managerial challenges to the engineering design community [4,5].

The three domains, shown in Fig. 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 [6–10]. Additionally, three decompositions co-exist: product decomposition into modules, process decomposition into design tasks, and organizational decompositions into design teams. If the decompositions are not well coordinated across the three domains, then inefficiencies may arise [11]. These observations are summarized as follows (note the areas numbered in Fig. 1):

Area 1: The organization is structured to “efficiently and effectively” deliver on a specific design (i.e., product architecture) [12].

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 materialize [13].

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 bidirectional communication among various development teams must be supported by the existing social network of the organization [14].

Product development decomposition creates 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 (design problem solving know-how), and deteriorated morale amongst designers. These are not hypothetical observations, but have been widely reported in various industrial settings.

Fig. 1 Model of product development
Including: automotive [15], aerospace [16], construction [17], electronics [18], and software [19]. There are several rigorous models that help mitigate these challenges and there is a pressing need by the PD community for an enterprise-wide model of PD information management [5,20].

In this paper, we propose a conceptual and computational framework that uses a multidomain approach to integrate the design team, the product architecture, and the processes (or tasks) carried out by the development organization. Thus, developing the basis for an enterprise-wide PD information management system that is capable of addressing the PD challenges outlined above. The framework we propose considers a sequential relationship: Data → Information → Knowledge → Product. We suggest that the steps taken by an organization to acquire data, to interpret information, and to apply knowledge will impact the final architectures of the firm’s products. The framework introduced in this paper is an attempt to understand some key issues in dispersed and collaborative PD as a result of the relational dependencies amongst the three PD domains. Analyzing these key relationships leads to two insightful research questions:

(a) Is there an insightful computational mapping between organizational social structures, processes, and product architecture? Can a network perspective achieve this?

(b) How is information and knowledge created and utilized in a PD organization? Can organizational knowledge be mapped to product architecture?

The rest of the paper proceeds as follows. In Sec. 2, we provide a literature review of publications discussing many analyses, theories, techniques, as well as results considering a multidomain approach to product development. In Sec. 3, we introduce the Data → Information → Knowledge → Product framework schema for studying the flow and transformation of data, information, and knowledge in the PD organization. Additionally, Sec. 3 proposes several constructs to operationalize the framework and “workout” some of the framework computational details. Section 4 discusses results for the ROBOCODE software case study. Finally, in Sec. 5, we discuss possible managerial benefits for the PD organization.

2 Literature Review

The development activities for complex products are typically decomposed into inter-related groups performing tasks in order to successfully develop a product. Consequently, product development is multifaceted endeavor requiring the choreographed relationship of many domains [21]. Eppinger and Salminen [8] discuss the possible relationship between the product, tasks, and organizational domains in product development. Sharman et al. [22] 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 design structure matrix (DSM) showing the relational arrangement of elements within the domains. Danilovic and Browning [23] propose a rectangular DSM construction relating different domains of the product development process. This new domain mapping matrix provides insights into the various characteristics of the product development process. It is worth noting that a general matrix mapping approach was discussed thoroughly by Yassine et al. [24]. Additionally, there is complementary research in process modeling, such as the process specification language and the business process diagram. These processes facilitate the exchange of process information among disparate systems [25,51].

Lindemann and Maurer [49] considered a multidomain approach that considers the complexity of multiple factors: including market complexity, product complexity, process complexity, and organizational complexity. They propose a scheme that relates these domains by the information-sharing activity taking place within an organization. These domain elements map to a new multidomain network. Gokpinar et al. [50] studied the relationship between the product architecture and the organizational structure by considering their joint impact on efficiency and quality in complex product development. They find that coordination problems within the development team, have a 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 [11] considers a mapping between a product, organization, and process by incorporating an affiliation network to measure a participant’s 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 [25] 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, by aggregating the task and resource networks, we can gain some understanding of which task require which resources as well as which resources are overutilized or underutilized. Shooter et al. [26] proposed 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 designed.

It appears that researchers predominantly view the relationship between product architecture and organizational architecture as starting with the product architecture and then working out an organizational design [28]. For example, Baldwin and Clark [27] derived the idea of a task structure matrix 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 organizational structure dictates the types of products designed by an organization. Sanchez and Mahoney [12] considered strategic decision making in a firm and how the organizational structure influences the available options of product architectural characteristics. Sako [29] 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. In contrast, other studies suggest that the product architectural choice of a firm influences the organizational design. These research findings suggest that there is a two-way relationship between product and organizational architecture [33,34].

Although there are studies in the literature, which consider the links between the many different domains (product, organization, task, etc.) in product development; what seems to be missing is a relational analysis and investigation of the mediating element or microlevel structure linking the domains together.

From the studies discussed above, it is clear that a simple decomposition of a product into its many subassemblies or functional components will neither have a one-to-one matching nor impact only a specific task. Furthermore, even when product development decomposition is worked out the results show some evidence of misalignment [9,33]. Consequently, there must be additional factors to consider in understanding the relationship between the product, the processes/tasks, and the organizational design.

3 Theoretical Overview

In this framework, we consider the mediating elements to be the creation and use of data, information, and knowledge within an organization. We represent these elements as network nodes. This representation provides a mechanism to study the
knowledge is a physical representation of principles and can further be defined as information with a high degree of certainty and validity. Knowledge is a conceptual model of principles that can lead to resolutions or actions.

Table 1  Summary of information science literature

| Data | —data are symbols organized according to the established algorithms [30]  
|      | —data are a series of disorganized facts and observations [31]  
|      | —data are convertible to information by analyzing, selecting, sorting, or other ways [30]  
|      | —data represent real facts [36]  
|      | —data can be quantified, measured, counted, and stored [34, 35]  
| Information | —information is the collocation of data—information is the end product of data processing [32]  
|      | —information is communicated and has relevance and meaning [33]  
|      | —information is the relationship between the inner arrangement of a systems [33]  
|      | —information is made up of more structured items [37]  
|      | —information is revealed or created each time data are interpreted successfully in the direction of increasing benefit and profit [38]  
| Knowledge | —knowledge is a physical representation of a product [34]  
|      | —knowledge has stability [34]  
|      | —knowledge has meaning and content assimilated for use to make decisions [35]  
|      | —knowledge is an emergent construct [39]  
|      | —knowledge is the sum or product of information [34]  
|      | —Knowledge can further be defined as information with a high degree of certainty and validity [43]  

One way to think of a product or a service is as the representation of knowledge into a usable and distributable form (i.e., products and/or services). To develop a product/service, 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 satisfy a 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 used in this paper borrows many of its mathematical concepts from graph theory and social network analysis [46, 47]. 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 PD, we consider four key networks: the team network (composed of individual development participants or teams), the process network (composed of tasks carried out by the organization), the physical network (composed of product components and subsystems), and the content network (composed of organizational databases). We can construct each of these networks by defining nodes (e.g., development participants in a team network or a physical component in a product network) and edges (relationships between actors) to represent a PD network, see Fig. 2(a).

Fig. 2  Network and matrix representation (a) five node network (b) corresponding five node binary matrix
The search themes in this example are adapted from the SIMA reference architecture Part 1: Activity Models [47], and Quality Characteristics and Metrics for Reusable Software [52]. The DB_j entry in the matrix is “1” if the database contains relevant data for a specific search theme or “0” if no relevant data is available within the database. This matrix provides the mapping between databases used by an organization and the search for data that is useful to projects undertaken by the organization. The search themes are based on the specific archetypes for a specific product development effort. In this case, we are discussing software; so terminology and archetypes for software are relevant. The column entry may change depending on the type of product being developed. For example, if developing an engine one entry in the matrix may be “piston design.” The organizational experts matrix [EP] represents the subject matter experts for a specific domain in the product development process. In this case, the individuals will be experts in the area of software development. Because this data sourcing is the result of individuals’ (within an organization) search schemes, acting on this layer is the social (or team) network, which we describe as a matrix [TM] and a query (or search) matrix [QR].

4 Proposed Framework

A conceptual model of the proposed framework is shown in Fig. 3. In this framework, we consider the product development process to be a sequence 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 [41]. However, their approach focuses on machine learning and organizational change. Alternatively, we study the PD process and attempt to frame the movement within and between layers using new ideas and definitions. Within this framework, we specify multiple layers and transformations. A key requirement is to appropriately define and characterize each layer and the relationships that exist between them. In this framework, we assume sequential flow between the layers. The definitions below will provide a more detailed exploration of these layers and transformations.

4.1 Data Layer. The data layer (see top of Fig. 3) consists of incongruent, nonaggregate, disconnected elements. This is considered as data collected independently by various functional units within an organization. We assume that each of these functional units are engaged in some form of environmental scanning that enables them to collect streams of data that are vital to their role within an organization. Sources of data may include journals, conferences, friends, individuals within an organization, professional contacts as well as many others [42].

Thus, the data layer consists of databases, described as a matrix [DB]. authored and organized from either internal or external sources. [DB] is an \( m \times n \) binary matrix, \( m \) represents the search themes that an individual uses or that exist within an organization when searching through a database.

Alternatively, to illustrate the relational structure of our model, we take advantage of the capabilities and flexibility of matrices to represent a network structure (see Fig. 2(b)). Matrices provide a consistent and tractable way to represent networks and to manipulate them using well-established matrix mathematical technique. Throughout this paper, we use the terms “matrix” and “network” interchangeably to mean or represent the same thing.

Additionally, to illustrate the relational structure of our model, we take advantage of the capabilities and flexibility of matrices to represent a network structure (see Fig. 2(b)). Matrices provide a consistent and tractable way to represent networks and to manipulate them using well-established matrix mathematical technique. Throughout this paper, we use the terms “matrix” and “network” interchangeably to mean or represent the same thing.

**Fig. 3** Data → Information → Knowledge → Product Framework (a) ROBOCODE Development Team (social matrix) [TM] (b) ROBOCODE database matrix [DB] (c) ROBOCODE expert matrix [EP] (d) ROBOCODE query matrix [QR]
length in the team network. This construct has its foundation in work by Conway [53], who suggests that the communication patterns of a design committee will determine the final system design. These communication patterns will be strongly correlated to dyadic and triadic relationships in the team network [21]. The NEI metric represents the “effort,” search cost, to find the appropriate information in the network to create knowledge. We use this metric as a proxy for the amount of time spent searching and integrating knowledge. For example, in the case in this paper the NEI value is 0.12, which suggests that 12% of the project time is focused on integration.

\[ \text{NEI} = \frac{\sum_{k=0}^{p} \sum_{k' \leq k} \text{TM}_{kk'}}{p(p-1)} \]  

(1)

If a network is completely connected, then each node is directly connected to all other nodes. In product development context, each node must have defined interfaces to interact with all others and a change at one node may impact every other node. During the development cycle, all nodes make design changes and, in theory, each node has to check that no integration problems have occurred. Since there is no “brokerage” level to manage the inter-connectivity (that is, there are no special “highly connected” nodes that can act as a “broker” [39,40], then this implies that majority of the project’s time is spent on integration. Alternatively, in the case of a chain, there is minimal time spent on integration.

4.2 Information Layer. We regard this layer as a composition of dyadic relations [40]. A dyad is represented by two nodes connected by a link. The justification for using a dyad is based on our earlier definition of information. Members in the organization will seek out one-on-one relationships 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. To illustrate the dyadic formations, consider a person who is a member of the marketing department that may have a relationship with someone in engineering and finance. This relationship could be due to one of three scenarios: (1) she/he sources data from engineering and finance databases, (2) she/he communicates directly, person-to-person, only within the social network, or (3) she/he sources data from databases as well as communicate person-to-person. If we consider such a network, it will consist of three nodes and two links, as shown in Fig. 5, which corresponds to two dyads. These two dyads represent two new information nodes.

To transition from the data layer to the information layer, we use the mapping function in Eq. (2), which captures the dyadic relationships in the augmented two-mode matrix. Thus, the new matrix \(Y\) represents the information network, which contains the frequency by which an individual or expert performs searches from a database and also communicates with another expert or individual in the organization. This construct builds on the algorithms from the PCANS model [24], which seeks to find second order relationships in a communication network. In our model, information is a second order construct. It builds a level of structure around the data to now become information. So, for each nonzero entry in matrix \(Y\), new information is created. Note that \([QR]^T\) represents the transpose of the query matrix and the “\(i\)” operator illustrates that the matrix is augmented. The resultant \(Y\) matrix for the ROBOCODE example is shown in Fig. 6. Inspecting the first row in the matrix (for “person 1”), the
matrix shows that “person 1” interacts (i.e., queries) with the “Code Builds” database twice, with “Patches” database twice, and with the “Bugs” database once. This row also shows that “person 1” interacts/queries persons 3, 4, 7, and 12. It is worth noting that while the [TM] matrix is considered to contain the general communication patterns; [Y] represents the communication patterns generated by searching for data for a specific product development effort, which may be different from the social construct suggested by [TM] [54].


From matrix [Y], we compute the information impedance (IIP) metric. IIP is a measure of the resistance to information formation as a result of an individual’s network centrality. This metric captures how easy (or difficult) it is to resolve queries. The more a person is queried, the less time she/he has to resolve each request because they become too busy. In practice, this individual represents the “go to” person in the organization because of their structural position in the network. Although this individual may be queried often, the more structural central he/she is positioned in the network there is a higher likelihood that their presence increases productivity and hastens the speed of which changes can be made within the networked organization because they will broker the relationship between unconnected individuals [55]. We subsequently use this metric for computing the knowledge-process matrix, [X], discussed later. Note the IIP calculations for the software example are in the last row of Fig. 6:

\[ IIP_k = \frac{\sum_{d=1}^{n} Y_{kd} \cdot SC_k}{SC_k} \quad \forall \quad n + 1 \leq d \leq n + p \]  

The denominator of Eq. (3) is defined as structural centrality (SC) and calculated using matrix [TM]. SC measures the control of communication flow in the network as a result of the individual’s or the database’s structural position in the network. We compute SC as the average of the degree, closeness, and betweenness centralities, respectively, for person k [40]. Other centrality measures could be included in calculating this average (e.g., culture centrality). The SC_k calculations for the ROBOCODE example are shown in the last row of Fig. 4(a).

\[ \text{Acting on the information layer (see Fig. 3) is the process relational network [PRR], the process responsibility network [PRM], and the information query network [IQM]. The [PRR] matrix is a square binary matrix that defines the dependency between of the various tasks in a project (e.g., project network). The [PRM] matrix is a binary matrix specifying which individual/expert is responsible for completing a specific task. The [IQM] matrix specifies search of specific knowledge-type resource by an individual. For example, given a set of tasks (captured in a [PRR] network) that must be completed for successful development, each individual assigned for performing a task as captured in [PRM] network) will source data from various sources (i.e., databases and/or experts) to acquire needed data to complete their tasks. The tasks, documented in the [PRR] network, provide the mechanism by which data are converted or transitioned into information by way of creating linkages between individuals in the network.}

As a result of the information query on the process matrices, we compute the process-search responsibility [RS] and process-search connectivity [PR] matrices. The purpose of these two matrices is to model the relationship between organizational knowledge and the search by the participants involved with the development project, who are responsible for completing given tasks. The [RS] represents the frequency of specific knowledge searches in order to complete a given process as a result of an individual task responsibility. [PR] represents the frequency of specific knowledge-type searches as a result of the interdependencies between the tasks. The [RS] and [PR] matrices are calculated as follows:

\[ [RS] = [PRM]^T[IQM]^T \]  
\[ [PR] = [RS]^T[PRR] \]  

4.3 Knowledge Layer. The transition from the information layer to the knowledge layer is the result of people in the social network sourcing various information nodes to answer questions regarding the project tasks (i.e., processes) and requirements that need to be addressed. Thus, the knowledge layer represents a set of connected blocks of information from the information layer. Since knowledge has a higher level of complexity than information, we assume that the knowledge layer consists of network relationships beyond the dyad, a triadic relationship. Knowledge is created when dyadic pairs interact [45].

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 → j and j → k, then there also exists i → k. We hypothesize that transitivity is a proxy for characteristics of repeatability, validity, and certainty in the information provided by a single member or group of members in a development team. In this framework, information will not transition to knowledge unless it is sourced or used frequently by others (i.e., high transitivity). Then, to transition from the information layer to the knowledge layer we must identify the triads in the information network. To define these triads, we consider the “ego” network for each team or individual in the PD organization [40]. A transitive triadic relationship consists of three nodes each connected to one another bidirectionally. In this framework, we are only searching for the ego network (e.g., nodes that are adjacent) for nodes that represent individuals (e.g., people). In Fig. 7, we show the ego networks for person 2 for the software-development...
example. In this case, the ego represents which databases or individuals are queried by person 2.

If we represent each node in Fig. 7 as "P" and consider the case for a transitive triadic relationship consisting of the three nodes Pi, Pj, andPk, then two nodes are connected if Pij = 1, Pjk = 1, or Pik = 1. For each node "P" in matrix [Y], we test this condition. Using matrix [Y] and the ego network representation, we construct the knowledge network/matrix. We use a mapping function (Eq. (6)), where matrix [Z] represents the knowledge network, P represents person and/or database.

\[
\text{Knowledge} = [Z] = \begin{cases} 
1 & \text{if } P_{ij} = P_{jk} = P_{ik} = 1 \\
0 & \text{otherwise}
\end{cases}
\]  

(6)

The resultant [Z] matrix for the ROBOCODE example is shown in Fig. 8. This matrix is a \((n+p)\) binary matrix. The value of “1” indicates that elements are connected in triads. Figure 9 shows the knowledge node created as a result of an exchange between person 1, person 2, and person 4. This is represented as knowledge, K2.

From the knowledge matrix [Z], we define the knowledge impact [KI] and the knowledge weight [KW] metrics. The knowledge impact metric is an assessment of the weighted impact of the databases and the individual experts on the organizational-knowledge content. The larger the impact value, the greater the impact of either the individual expert or the database on the organization’s product development activity. The knowledge weight metric is an assessment of the importance of a particular knowledge (node) on the organization. This metric is subsequently used in determining the knowledge-to-architecture computations. The larger the knowledge weight value, the greater the influence of the individual expert or the database content. Note that the KI and KW calculations for the ROBOCODE example are shown in the last row and column, respectively, of the matrix in Fig. 8.

\[
\text{KI}_d = \sum_{s=1}^{a} Z_{sd} \quad \forall \quad 1 \leq d \leq n + p
\]

(7)

\[
\text{KW}_s = \sum_{d=1}^{n+p} \text{KI}_d Z_{sd} \quad \forall \quad 1 \leq s \leq a
\]

(8)

Additionally, we compute the knowledge-process matrix [X]. This matrix provides the relationship between searching and subsequent use of knowledge when completing the required tasks.

\[
[X] = [Z][PR]
\]

(9)

Because the knowledge layer consists of highly reliable information elements, we assume that this layer represents organizational “know-how.” The challenge is to successfully map this knowledge to functional units and subsequently reduce the decomposed modules to physical elements of a final architecture. We assume that each knowledge construct that is created within an organization may have an impact on overall product development. Moreover, we assume that network search cost will further influence the impact of knowledge, by limiting knowledge acquisitions by the individuals in the PD organization. Thus, we use
information impedance of the network as a multiplier. Information impedance represents the nodal constraints of the network that impedes the formation of triadic knowledge nodes from dyadic information nodes. We apply information impedance to the knowledge-process matrix $X$, in order to realize a binary knowledge-process matrix $XX$, which models the relationship between the search for and the subsequent use of knowledge when completing the required tasks. The transformation from $X$ to $XX$ is realized using the following procedure:

**Procedure A:**

1. Multiply all the entries in matrix $X$ by the IIP metric of Eq. (3).
2. Compute the median for all the entries in the matrix. This is justifiable because we know that the search cost is 

Fig. 8 ROBOCODE knowledge matrix $[Z]$, knowledge impact $[KI]$, and knowledge weight $[KW]$ (computed using Eqs. (6)–(8), respectively)
The relationship between the processes and the constraints.

As we transition from the knowledge layer to the architecture layer there are additional interactions to consider. Acting on the knowledge layer is the constraint network $[\text{CF}]$ that specifies the relationships between the constraints that govern the physical design or product function, and the constraint search network $[\text{CS}]$ that identifies the search scheme used by individual(s) as they attempt to tackle problems imposed by the constraints. Finally, we have a process-constraint network $[\text{PC}]$ that defines the relationship between the processes and the constraints.

We define a knowledge constraint matrix $[M]$ as in Eq. (10), where this matrix specifies the frequency by which a specific knowledge node was referenced when addressing a specific constraint.

$$[M] = [XX][PC]^T$$

When successful combinations of knowledge elements and groupings have been found the elements are now decomposed into various chunks (product architecture layer). To transition from knowledge (cognitive domain) to the product (physical domain), we define the relationship between the knowledge and the architecture layers. We use a mapping function that considers the knowledge in the network and the search behavior of the individuals in the network as they attempt to address the various constraints.

The architecture ratio, $AR$, quantifies the relationship between the available knowledge in the network and how it is accessed by the PD team. The construct we use borrows from text retrieval models observed in search engine design [56]: namely, the vector space model which is used to represent items in a collection. These items may be search terms or documents being searched. For example, a document collection composed of $n$ documents that are indexed by $m$ terms can be represented by an $m \times n$ term-by-document matrix $[A]$. Then, matrix elements $a_{ij}$ represent the frequency at which term $i$ occurs in document $j$. Query matching in the vector space model can be viewed as a search in the column space of $[A]$. The column space is basically a subspace spanned by the document vectors. A measure commonly used for querying is cosine $\theta$ as shown in Fig. 10 [56]. We adopt this well-known measure for modeling an organization’s ability to address constraints. In our model, the PD team uses knowledge within the network to address design constraints. There will be some design limitations based on mechanical assemblies and engineering requirements [46].

When the $AR \geq 1$, this suggests that there is sufficient knowledge in the organization to address the constraints; thus, leading to a suitable product. However, if $AR < 1$, this suggests that knowledge is insufficient or not utilized efficiently. As cosine $\theta$ approaches 1, then $\theta$ approaches 0. This behavior suggests that an organization’s ability to address the constraints increases. Thus, $AR$ is computed as follows:

$$[AR]_{ik} = \frac{[AR1]_{ik}}{[AR2]_{ik}}$$

Equation (11) is the cosine $\theta$ computation for the individual search scheme and the knowledge constraint matrix, where $||M||$, $||CS||$, and $||CF||$ are the Euclidean of matrices $[M]$, $[CS]$, $[CF]$, respectively. The numerator computation reflects an individual’s search to find knowledge in the network. The denominator computation reflects an individual’s search behavior to account for and to address the dependencies between the constraints. The relationship ($AR \geq 1$) is tested for each $AR$ entry. This testing allows identification of knowledge nodes that satisfy the ratio. The knowledge nodes that satisfy ($AR \geq 1$) are subsequently placed into knowledge groups (e.g., modules), as described next.

4.4 Product Architecture Layer. The product architecture layer represents a chunking of 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. However, the
mapping of the knowledge elements to functional chunks is non-trivial. For example, in considering a mechanical system or device and perhaps some software systems, there exist some physical or implementation connectivity limitations [46]. Such limitations will determine the scope of architectures that can be realized. To illustrate this concept, consider Fig. 11 which represents a partial knowledge network. This example is comprised of five knowledge elements (K1, K2, K3, K4, and K5). For example, we can consider K1 and K2 to represent knowledge about database design, K3 and K4 to represent knowledge about database synchronization, and K5 to represent knowledge about database content retrieval.

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 the previous discussion, that acting on the knowledge layer is a configuration network, which determines possible architectural choices. These architectural choices depend on connectivity of the design constraints. This represents the final transformation—from knowledge to product. The product network is the aggregation of knowledge nodes. To combine the knowledge nodes and create the knowledge grouping matrix $[KG]$, we use the “architecture ratio” values. The computational methodology is as follows:

**Procedure B**

2. Multiply $[AR]$ by the $[KW]$ obtained from Eq. (8).
3. Calculate the NEI using Eq. (1).
4. Calculate a lower network effect boundary (LNEB) and upper network effect boundary (UNEB) for the knowledge-weighted Architecture_Ratio, as follows: These boundaries are equivalent to confidence intervals used in statistical computations.

$$LNEB = (1 - \text{NEI}) \left( \frac{(\text{Average Architecture Ratio}) \times (\text{Knowledge Weight})}{2} \right)$$  

$$UNEB = (1 + \text{NEI}) \left( \frac{(\text{Average Architecture Ratio}) \times (\text{Knowledge Weight})}{2} \right)$$  

The “Average Architecture_Ratio” is defined for each knowledge node in $[AR]$ matrix averaged over all the individuals.

5. Group knowledge elements whose Architecture_Ratio fall within overlapping LNEB and UNEB values.
6. The final outcome is the knowledge grouping matrix $[KG]$.

The $[KG]$ matrix represents the knowledge network for a development process shown in Fig. 12. From this network, the knowledge architecture can be derived based on the chunking of knowledge elements. Consequently, this knowledge architecture can either be mapped to product architecture for a new product, or compared to existing product architectures in order to recommend architectural improvements. These improvements originate from noted discrepancies between the derived knowledge network and the existing product architecture network.

### 5 Software Case Study

For demonstrating the methodology and computations proposed in this framework, we choose the open-source project, Robocode. An open-source software (OSS) development project provides an excellent opportunity to work through this framework because OSS development projects exploit the distributed intelligence of participants across the world [57]. Using this case, we investigate how software requirements and the social structure shape the communication patterns among the developers, and subsequently impact the overall product architecture of the software source code.

In collecting the data, we used log files from the Robocode OSS project, as well as general software-development procedural manuals. We construct the software source-code network (i.e., the product architecture network) at the package level using Lattix [38]. For performing the various network (i.e., matrix) calculations and analyses, we used UCINET [48]. The relationships

---

7The clustering algorithm of Yu et al. [44] is used to identify the knowledge architecture and resultant knowledge modules.

8Robocode is an educational robot design game (http://robocode.sourceforge.net).
between the packages are based on function or method calls between the files. The programmer social network is based on who communicates with whom via logs regarding code updates and changes to the same source files (e.g., packages or class files). We make the assumption that the source-code network will be impacted by the sharing of code among the programmers, as well as by how efficient the programmers are at searching the source-code network and then interpreting and implementing this new knowledge into the software development. Finally, the search schemes used in the case study are adapted from general guidelines in product development [47].

Using this framework, we can create a knowledge network based on knowledge search and acquisition to address various constraints imposed by the organization while developing the ROBOCODE software. The knowledge elements (i.e., nodes in this knowledge network) are the results of the communication and interaction between the developers as they attempt to solve problems during the development project. The knowledge network captures these interactions explicitly, which supports traceability from the organizational-knowledge sources to the developers and databases (i.e., which developers and databases are used to create a specific knowledge node). For example, knowledge node K35 in Fig. 12 is represented by BU-P5-P11 node in Fig. 13, which means that the organization can map both the interaction between development team members as well as their querying behavior. By documenting these details the organization can better understand how the interactions amongst their resources impact knowledge and product architectures. Additionally, this network allows the PD organization to understand which classes, packages, functions, or individuals are driving the overall development as a result of the product requirements, constraints, and social interactions.

The knowledge network, which represents the knowledge content for the ROBOCODE project, is shown in Fig. 12. Although there were originally 45 knowledge nodes (see matrix [Z] in Fig. 8), 33 nodes were useful in addressing the constraints during the ROBOCODE software-development project. This suggests that a group or an organization has more knowledge available than what will be used on a given project.

The knowledge nodes in Fig. 12 map to modules and components in the physical and tangible domain. The question is how this mapping might take place. To understand this mapping let us consider the PD process in light of the ROBOCODE project. During the system-level design phase, the system designer decides to decompose the ROBOCODE software into eighteen modules (see the list of JAVA package files in the right-hand side of Fig. 13). Within this open-source structure, there is no formal team assignment. We assume that as the PD process transitions to the detailed design phase, programmers migrate to their areas of expertise. As they source information to develop their code, new knowledge is created within the project that has a mapping to the 18 JAVA package files decided by the system designer. In the left-hand side of Fig. 13, we have identified nine modules (using the clustering algorithm cited in footnote 6) based on the knowledge used and created during this design project.

Figure 13 shows a hypothetical mapping between the derived knowledge modules and the existing package files. The mapping scheme in practice is primarily subjective. This subjectivity is based on the understanding of general software-development protocols and procedures. These protocols and procedures suggest
which packages will require what types of knowledge, thus the possible interaction between the software developers working on the project. Using this subjective mapping approach and considering the general protocols in software development, we suggest a mapping between the knowledge nodes and packages files as shown in Table 2. For example, module 1 (in the knowledge architecture) can map to ‘battlefield’ and ‘battle’ package files (in the actual software architecture). This mapping is performed based on firsthand knowledge of what nodes (from the knowledge network) will be required to code these two package files. The rest of the mappings in Table 2 are performed using a similar approach.

6 Summary and Conclusion

In this paper, we propose a novel framework that encapsulates a multidomain approach to understanding the product development process; it operationalizes the development of a product architecture. This framework suggests that a product is the final outcome of the transformation from data-to-information-to-knowledge. A unique feature of this framework is that it provides a way to represent the knowledge nodes as elements of the databases search and the people involved. This may suggest a mechanism for tracking knowledge creation and knowledge use within the PD organization. A summary of the required inputs and computations is shown in the flow chart of Fig. 14.

The proposed framework does not provide a specific final product architecture; however, it provides a detailed view of how organizational knowledge and product development activities may impact the knowledge architecture and consequently the product architecture. The framework provides the means of supporting a view of products as a “bundle of interactions and knowledge.” This lays the ground work for better understanding the impact of architecting decisions on the overall product development process. The final outcome of this framework is that it provides a mechanism for “knowledge combinations” which subsequently provides a way to establish a relationship between organizational knowledge and product architecture.

The proposed framework represents an early-stage attempt at building a formal understanding (theory and tools) of the complex dynamics of distributed, collaborative product development. Because this framework is grounded in network science it integrates the interdependencies between the various PD domains and bridges the gap between a technical and a social treatment of the product development activity. This provides an integrative model for understanding the “complete” PD system.

<table>
<thead>
<tr>
<th>Knowledge module</th>
<th>Knowledge nodes</th>
<th>Package files</th>
<th>Package file descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1</td>
<td>K20, K30, K44, K45</td>
<td>Battlefield, battle</td>
<td>Handles all battlefield design</td>
</tr>
<tr>
<td>Module 2</td>
<td>K9, K14, K32</td>
<td>gfx, battleview</td>
<td>Handles graphics and view design</td>
</tr>
<tr>
<td>Module 3</td>
<td>K19, K28</td>
<td>Manager, security</td>
<td>Handles the messaging with the battle environment</td>
</tr>
<tr>
<td>Module 4</td>
<td>K3, K7, K11, K25, K36, K42</td>
<td>Editor, text</td>
<td>Handles the sound files and widgets</td>
</tr>
<tr>
<td>Module 5</td>
<td>K4, K6, K10, K13, K17, K38, K43</td>
<td>Sound, dialog</td>
<td>Handles the sound files and widgets</td>
</tr>
<tr>
<td>Module 6</td>
<td>K5, K29, K29, K39</td>
<td>Exception, io, peer</td>
<td>Handles the sound files and widgets</td>
</tr>
<tr>
<td>Module 7</td>
<td>K35</td>
<td>Main, control</td>
<td>Entry point into the Robocode Software, initiates program</td>
</tr>
<tr>
<td>Module 8</td>
<td>K15, K23</td>
<td>Util, packager</td>
<td>Discretizes the packages for ease of use</td>
</tr>
<tr>
<td>Module 9</td>
<td>K24, K27, K40, K41</td>
<td>Repository</td>
<td>Manages overall project files</td>
</tr>
</tbody>
</table>

Fig. 14 Summary input–output flow chart
To walk through the mechanics of the proposed framework (i.e., the network analysis computations and matrix mathematics), we have used a case study that involves the PD process for the open-source software ROBOCODE. Analysis of this case study show that it is possible to map the derived knowledge network to a new (or existing) product architecture. Although the mapping illustrated in the case study is subjective, it does suggest a realistic approach, but at the same time calls for further research for developing formal mapping procedures.

The proposed framework was implemented on two case studies [59]. A jet engine PD project and a software PD project. In both cases, the calculations were performed using Microsoft Excel (to manipulate the matrices) in conjunction with UCINET (to calculate the various network properties). Our experience with both of these cases (considering their differing sizes and applications) confirm that the computations required for implementing the proposed framework scale up well for large problems and different types of applications.

Although we believe that this framework does get to the heart of studying product development in terms of social attributes as opposed to only technical constraints, there are elements of PD that are strictly technical. We have attempted to account for this by modeling the technical constraint relationships. Additionally, we consider that data, information, and knowledge are by-products of social interactions under certain conditions (e.g., tasks and constraints). Although we support this assumption with current information science literature, there may be certain factors that are not properly captured in the simplification to these social interactions only. For instance, we could attempt to model data, information, and knowledge as other types of networks motifs (i.e., diads, triads, or higher order configurations) in order to compare alternative modeling approaches.

Another limitation is the acquisition of the input parameters. In theory, each input parameter can easily be approximated or gleaned from PD logs and PD handbooks in the organization. In the presented case study, we utilized general development guidelines from NIST and previous open-source software-development studies to parameterize the model. However, details of data acquisition for real world PD project merits further research and investigation.

Finally, it is worth noting that the networks (and matrices) utilized in this model might not be able to represent the dynamic nature of relationships during a PD process. That is, it possible for a triad to be found that is based upon network connections that are actually not related because they occurred at different times in the PD process for two unrelated tasks. Perhaps this is an additional level of complexity reserved for future work.

Despite these limitations, this framework does represent a level of granularity that provides new insight into the mediating links of data, information, and knowledge that may shape 21st century product development [60].

Appendix: Nomenclature (Items Highlighted in Gray are Input Data)

\[i: \text{Index for databases used by the individuals in an organization for searching, where } 1 \leq i \leq n\]

\[j: \text{Index for search themes (i.e., subjects) used when searching a database, where } 1 \leq j \leq m\]

\[k: \text{Index for individuals that are members of the organization as well as considered experts in specific search themes (i.e., subjects), where } 1 \leq k \leq p\]

\[d: \text{Index for combined databases and individuals in a PD project, where } 1 \leq d \leq n + p\]

\[b: \text{Index for tasks (i.e., processes) that must be completed in a PD project, where } 1 \leq b \leq v\]

\[c: \text{Index for constraints that must be addressed by the PD organization, where } 1 \leq c \leq q\]

\[s: \text{Index for the number of knowledge nodes, where } 1 \leq s \leq a\]

\[z: \text{Index for the number of modules in the final product architecture, where } 1 \leq z \leq y\]

\[DB: \text{Database matrix } (m \times n) \text{ binary matrix, where } DB_{ij} = 1, \text{ if database } i \text{ contains relevant data for search theme } j; \text{ and } 0 \text{ otherwise}\]

\[EP: \text{Expertise matrix } (m \times p) \text{ binary matrix, where } EP_{jk} = 1, \text{ if individual/expert } k \text{ contributes relevant data for search theme } j; \text{ and } 0 \text{ otherwise}\]

\[QR: \text{Query matrix } (m \times p) \text{ binary matrix, where } QR_{jk} = 1, \text{ if individual } k \text{ searches for relevant data using search theme } j; \text{ and } 0 \text{ otherwise}\]

\[TM: \text{Social network matrix } (p \times p) \text{ binary matrix that maps the communication patterns between individuals within an organization. Entries in the matrix are } "1" \text{ if two individuals communicate/interact; and } "0" \text{ otherwise}\]

\[Y: \text{Information matrix: Augmented } p \times (n + p) \text{ matrix that contains the frequency by which an individual/expert performs searches from a database and also communicates with another expert (i.e., dyadic relationships)}\]

\[Z: \text{Knowledge matrix, } a \times (n + p) \text{ binary matrix. It shows the number of transitive triadic relationships}\]

\[PRR: \text{Process relational matrix } (v \times v). \text{ A square binary matrix that defines the dependency between of the various tasks in a project (e.g., project network). An off-diagonal element in this matrix is equal to } "1" \text{ if tasks are dependent; and } "0" \text{ otherwise}\]

\[PRM: \text{Process responsibility matrix, } (p \times v). \text{ A binary matrix specifying which individual/expert is responsible for completing a specific task. The PRM}_{ik} \text{ matrix entry is } "1" \text{ if an individual } k \text{ has some responsibility in completing task } b; \text{ and } "0" \text{ otherwise}\]

\[IQM: \text{Information query matrix, } (n + p) \times p \text{ matrix that specifies search of specific knowledge-type resource by an individual. The IQM matrix entry is } "1" \text{ if an individual searches an organizational resource (either a database or another expert); and } "0" \text{ otherwise}\]

\[RS: \text{Process-search responsibility matrix, } v \times (n + p) \text{ matrix that represents the frequency of specific knowledge searches, by individuals, in order to complete a given task. It represents the use of knowledge that depends on resource allocation}\]

\[PR: \text{Process-search connectivity matrix, } (n + p) \times v \text{ matrix that represents the frequency of specific knowledge searches as a result of the interdependencies between the tasks}\]

\[X: \text{Knowledge-process matrix. A valued } (a \times v) \text{ matrix that provides the relationship between the searching and subsequent use of knowledge when completing the required tasks}\]

\[XX: \text{Recoded knowledge-process matrix. A binary } (a \times v) \text{ matrix. This reduces the original valued matrix } [X] \text{ to a binary matrix } [XX]\]

\[CF: \text{Constraint matrix, } (q \times q) \text{ binary matrix that contains organizational constraints regarding specific capabilities or engineering constraints and their relationships. An entry in the } [CF] \text{ matrix is } "1" \text{ if two constraints interact or have some dependency; and } "0" \text{ otherwise}\]

\[PC: \text{Process-constraint matrix, } (q \times v) \text{ binary matrix that maps the relationships between processes and the constraints that must be addressed. The } PC_{cb} \text{ entry is } "1" \text{ if a process requires that a specific constraint be addressed, else } "0" \text{ otherwise}\]

\[CS: \text{Constraint search matrix, } (p \times q) \text{ binary matrix that specifies the individual’s searches to address various constraints that must be addressed in the project. The } CS_{bc} \text{ entry is } "1" \text{ if the individual searches the organization’s resources to address a specific constraint, else } "0" \text{ otherwise}\]

\[M: \text{Knowledge constraint matrix, } (q \times a) \text{ matrix that specifies the constraints that must be addressed. } a \text{ specifies the}}
available knowledge within an organization. This is a valued matrix where each $M_{es}$ entry specifies the frequency by which a specific knowledge node was referenced when addressing a specific constraint

$[KG]:$ Knowledge grouping matrix. $(a \times y)$ matrix which corresponds to the product architecture

NEI: The network effort index is an index used for estimating effort or search cost required to combine or to group information into knowledge nodes. Effort is a cognitive measure based on the network overall structure

IIP: Information impedance is a measure of the resistance to information formation as a result of an individual’s network centrality

AR: Architecture Ratio quantifies the impact on product architecture as a result of organizational knowledge and constraints that must be addressed

KI: The knowledge impact metric is an assessment of the weighted impact of the databases and the individual experts on the organizational-knowledge content

KW: The knowledge weight metric is an assessment of the importance of a particular knowledge node on the organization

References


