Generalized Design Knowledge and the Higher-Order Singular Value Decomposition

Andy Dong and Somwrita Sarkar
University of Sydney, Australia

The question of what constitutes generalized design knowledge is central to design cognition. It is knowledge of a generic variety about products, the type of knowledge that is not principally related to any one product *per se*, but to knowledge about a broad class of products or a broad class of operations to produce products. In this paper, we use a complex systems perspective to propose a computational approach toward deriving generalized design knowledge from product and process representations. We present an algorithm that produces a representation of generalized design knowledge based on two-dimensional and multi-dimensional representations of designed objects and design processes, with the objects and processes being represented as a complex network of interactions. Our results show that the method can be used to infer and extract macroscopic, system level organizational information, or generalized design knowledge, from microscopic, primary or secondary representations of objects and process.

Introduction

'Generalized design knowledge' is knowledge of a generic variety about products, the type of knowledge that is not principally related to any one designed object *per se*, but to the knowledge about a broad class of designed objects or a broad class of operations to produce those objects. Such knowledge is needed to achieve innovation [23]. The question that this paper raises is how this generalized knowledge can be efficiently calculated and represented from experience gained through interaction with design representations. Stated another way, what is the representation for this generalized design knowledge?

James Perner's three-stage model for the cognitive development of representation skills is a useful starting point to address this question [19].

According to Perner, children progress through at least three stages in the development of cognitive skills associated with mental representation. Initially, children have the ability for *primary representations*, which have a semantic fidelity with the external world. Children can represent that an apple is an apple, but not that a computer named Apple is not the same as the apple that can be eaten. In the next stage, children develop the capacity for *secondary representation*. These are models of the referent object, but they may not necessarily have a direct semantic relation to referent object. At this stage, children can represent that an Apple computer is not the apple fruit, but could engage in pretend play to represent any object as an Apple computer or an apple fruit. Secondary representations are fundamental to the cognitive design skill of 'seeing as' because they allow us to entertain multiple models and interpretations of intermediate design representations. They allow us to have a model of what the world could be like, rather than as it is, which is essential to design. As Perner explains, "secondary representations are purposely detached or "decoupled" from reality. They are at the root of our ability to think of the past, the possible future, and even the non-existing, and to reason hypothetically." [19, p. 7]

However, one more stage of development in representational abilities is needed: the *tertiary* ability for *meta-representation*. Meta-representation is, literally, a representation of a representation. We start to understand that there is an ascribed relationship between the referent object and our representation or symbol for it, and we represent this relationship, which Perner names the 'representational relation'. Changing this relation is at the basis of using the same symbol to represent multiple objects and their respective semantic (mental) representations (or equivalently using different symbols to represent the same object). A symbol or the set of symbols in the representational relation can actually be an aggregate over multiple simpler perceptual or conceptual experiences.

As we represent the relation between the referent and its (mental) representation, we gain a capacity to produce alternative interpretations of the referent object by modifying the meta-representation. Stated in a more colloquial way, we can decide to change the way we conceive of an object. We can change our mental representation not by arbitrarily changing it *per se*, that is, by deciding that an apple is actually an orange, but by updating our representational relation, such that the mental representation now becomes consonant with the way that we represent the relation (e.g., as a fruit or as a computer). This skill allows us to compare alternative design concepts and is at the root of developing alternative representations [18].

Having the capability for meta-representation thus means having the representation of the concept of representation itself. It lies at the core of developing generalized design knowledge as represented in primary representations, because meta-representation entails constructing and having a representation for a model of previous and current design knowledge. The issue that this paper takes up is to develop a formalism to compute this generalized design knowledge. If a system has the capacity to infer or extract generalized design knowledge, then it should be able to group designed products and design processes into plausibly coherent categories. This paper will show how the singular value decomposition (SVD), and its more general form, the higher-order singular value decomposition (HOSVD), provides a formal way to produce generalized design knowledge as it is embodied in primary design representations.

While the literature on finding structure in data is both vast and deep in the machine learning literature, all approaches begin from either a heuristic about the expected clustering of the data or some pre-defined metric of similarity. More recent approaches to find structure from data start from a more principled cognitive science basis. For example, Kemp and Tenenbaum propose an algorithm which starts from a set of known grammars of knowledge structures, and then fits the structure to the data with the highest *a priori* probability [14]. Our aim is not to find the 'right' data structure for design representations, but rather to identify a way to make sense out of a set of design representations that may, on the surface, appear to be unrelated. Second, while we will not show it in this paper, our further aim is to produce plausibly correct structures, reflecting the observation that designers can identify multiple plausible concepts in design representations rather than a single *a priori* correct representation. In other words, we are aiming to produce suitable 'concepts' [22] from existing design representations. Design representations may contain both similarity relations, i.e., products that are similar to other products, and relational data, relationships between components in a product. The fundamental problem for designers is what knowledge from one domain can be applied in another domain or what exemplars [6; 27] provide broad knowledge frameworks for another domain [29].

Modeling Design Representations as Matrices and Tensors

To develop a representation for computing generalized design knowledge from primary and secondary design representations, we will employ a complex networks perspective. Any complex system can be viewed as a complex network using a graph structure, where the nodes are the system elements and the links are structural, behavioral, functional, or any other type of interaction relationship between the nodes. A matrix, where the rows and columns represent nodes and the matrix entries represent these relations, can fully describe a graph. Thus, a graph and its corresponding matrix representation are equivalent representations of the same system.

A primary advantage of adopting this formalism is that it allows us to represent the entire system from the microscopic level (primary and secondary design relations) and simultaneously allows us to analyze, infer, and extract macroscopic level information at the system level (generalized design knowledge). The basic premise in complex systems studies is that interactions at the microscopic level often produce emergent macroscopic effects that cannot be inferred from an analysis of the microscopic relations. Since generalized design knowledge develops over time, over multiple design experiences, and over interactions with multiple designed objects, it will have this emergent quality of complex systems.

The essentials of the mathematical structure are as follows. A designed object, product, family of products, or a set of design processes is represented as matrices (or equivalently, graphs). Linear algebra algorithms, specifically singular value decomposition (SVD) and higher-order SVD (HOSVD), which are matrix decompositions that produce optimal lower-dimensional descriptions of higher order data, are then used to infer generalized design knowledge from the matrices of primary and secondary design information.

We will begin the discussion with an analysis of representations of designed objects rather than design processes, but the mathematical representations of both are entirely homologous. For simplicity, let us begin in a two-dimensional space wherein a designed object or a product has n functions and m behaviors associated with it. This is the type of representation used in the market analysis of products, wherein a product is represented simply as a bundle of attributes. It is a common representation in systems engineering [4], where a product, process, or organization can be represented as a bundle of functions that are fulfilled by performing certain behaviors, or a set of behaviors that are fulfilled by structural components. We can now represent a product as an $n \times m$ matrix $\boldsymbol{X}$ (or equivalently a bi-partite graph) where n is the number of functions and m is the number of behaviors. Thus, each entry $x_{ij} = 1$ if a product fulfills a function i, $i = 1$ to n with behavior j, $j = 1$ to m. The addition of another dimension, the structural one, will produce an order 3 tensor that can represent, for one product, the entire Function-Behavior-Structure (FBS) framework [11]. The tensor element $x_{ijk} = 1$ will signify if a product has a structural component k that takes a part in fulfilling function i through behavior j. For now, we will concentrate on the 2D matrix representation, and develop the tensor representation later.

This is a simplified, idealized representation of a designed object, but nothing in this analysis hinges on whether this two dimensional representation is "true". Different designers can come up with different, equally valid matrices for the same object. They need not even be the same size, and different sized matrices, representing the object at different levels of granularity, may be developed. Indeed, this kind of parallel, equally valid, multiple representations lie at the basis of producing generalized design knowledge. The mathematics of analysis, therefore, must necessarily be general and not specifically tied to any particular or typical object.

Thus, our mathematical analysis is general and not tied to a specific model. A similar form of representation has previously been used to transform complex design optimization problems represented in symbolic equation form into a matrix format [21]. Although the matrix $\boldsymbol{X}$ is order 2 (the number of dimensions), the product representations form a vector space of dimension nm. This is because each row is a vector in $\Re^n$ and each column is a vector in $\Re^m$. Thus, the dimension of the vector space of all $n \times m$ matrices will be the number of matrices needed to define a basis for this space, which is nm.

For example, consider a product with 3 functions and 2 behaviors. Then we have a 3×2 matrix describing this product. Fig1 shows the basis for all 3×2 matrices. Each matrix shows 1 function or 1 behavior. Now consider a product that has functions 1 and 2 but not 3, and behaviors 1 and 2 but not 3. Obviously, just 4 of the basis matrices, and not 6, can represent this product. Further, the resulting matrix will be $\boldsymbol{X} = [1\ 1\ 0;\ 1\ 1\ 0]$. Clearly, for both n=2 and m=2, a lower order, 2D, representation can capture the full design knowledge. Thus, we do not need all the 6 dimensions to describe the knowledge of the product. In many cases, a lower-dimensional representation will be sufficient to capture the full design knowledge. As we will see, a lower-dimensional representation is not only sufficient to capture generalized design knowledge, it is also necessary to bring out the hidden patterns and relationships that are latent in the representation, and are not explicit in the original representation. We now show that the singular value decomposition (SVD) generates an optimal lower order representation by generating an orthonormal basis for an $n \times m$ matrix.

$$\begin{bmatrix}1&0&0\\0&0&0\end{bmatrix} \quad \begin{bmatrix}0&1&0\\0&0&0\end{bmatrix} \quad \begin{bmatrix}0&0&1\\0&0&0\end{bmatrix}$$

$$\begin{bmatrix}0&0&0\\1&0&0\end{bmatrix} \quad \begin{bmatrix}0&0&0\\0&1&0\end{bmatrix} \quad \begin{bmatrix}0&0&0\\0&0&1\end{bmatrix}$$

Fig1. Basis space for a product with 3 attributes and 2 functions

It is well known that a vector space can be best represented by an orthonormal basis. An orthonormal basis is a set of linearly independent, mutually orthogonal vectors of unit length. Any other vector in that space can be expressed as a linear combination of the orthonormal basis. If we could find such an orthonormal basis, this can provide a model for $\boldsymbol{X}$, because the basis allows us to express all possible design representations as linear combinations of the basis vectors. Further, as we show later, we can also decide how many orthogonal vectors, i.e., how many dimensions, are necessary to optimally describe a specific product. There is such a method to find the orthonormal basis satisfying these requirements. For any arbitrary rectangular matrix, we are guaranteed to find the orthonormal basis using the singular value decomposition (SVD) [26].

The singular value decomposition is a matrix factorization technique that calculates a matrix $\boldsymbol{X}$ as a product of three other matrices, such that $\boldsymbol{X} = \boldsymbol{USV}^{\mathrm{T}}$. By definition, $\boldsymbol{U}$ is an $n \times r$ matrix that is an orthonormal basis for $\Re^n$, and $\boldsymbol{V}$ is an $m \times r$ matrix that is an orthonormal basis for $\Re^m$ where r is the rank of $\boldsymbol{X}$. The vectors of $\boldsymbol{U}$ are called the left singular vectors, the vectors of $\boldsymbol{V}$ are called the right singular vectors. The $r \times r$ diagonal matrix $\boldsymbol{S}$ contains the singular values that contain the scaling information on how a vector is transformed when it goes from $\Re^n$ to $\Re^m$ and are usually arranged in a decreasing order of magnitude. The number of singular values is equal to the rank r of the matrix $\boldsymbol{X}$. Thus, SVD re-represents the original matrix in a diagonalized form, and produces a new derived independent set of basis vectors from the original information. Note that the vectors in the original $\boldsymbol{X}$ can now be represented as a linear combination of the vectors in $\boldsymbol{U}$ and $\boldsymbol{V}$, their components multiplied by the singular values in $\boldsymbol{S}$. Since the singular values are arranged in decreasing order, the first k largest singular values are the most important in capturing information. For example, for our earlier example from Figure 1, performing an SVD for $\boldsymbol{X}$ = [1 1 0; 1 1 0] will show that only two singular values and the corresponding singular vectors are needed to describe the original matrix in full

The main outcome of SVD is that the relation between the designed object and its representation $\boldsymbol{X}$ can now be represented in an abstract form as vectors that are linear combinations of derived orthogonal bases and singular values. In summary, the orthonormal bases and the singular values provide a model for $\boldsymbol{X}$, that is, a meta-representation in Perner's model.

This concept scales to higher-order (order of $\boldsymbol{X} > 2$) product representations, since a representation of product knowledge with the FBS ontology [11] or with functional modeling (product, function, and flow) [24] would make $\boldsymbol{X}$ order 4 and 3, respectively. Consider our earlier discussion on representing, for example, each product as a matrix of functions and behaviors. If multiple products, each represented as a matrix, are "stacked" as "slices", then we have a 3D or 4D "cube" to represent a family of products in terms of their functions and behaviors.

Matrices with order greater than 2 are generally called tensors, and, following the accepted notation, tensors are represented with Euler capitals, e.g., $\mathscr{X}$. The decomposition of a tensor into derivational modes (equivalent to the left and right singular vectors in SVD) and a core mode that shows the level of interaction between the components in the derivational modes was first proposed by Tucker in order to handle multi-dimensional psychological data [28]. Tucker solved this composition for an order 3 tensor, and the solution was recently extended it to N modes by De Lathauwer et al. [7], who showed that the Tucker decomposition is a generalization of the singular value decomposition. In so doing, they named the Tucker decomposition in N modes the higher-order singular value decomposition (HOSVD). Like SVD, the HOSVD decomposes a tensor $\mathscr{X}$ into a core tensor $\mathscr{G}$ (equivalent to the matrix of singular values **S**) and a set of matrices **A** (equivalent to the left and right singular vectors **U** and **V**) along each mode of $\mathscr{X}$. For a tensor $\mathscr{X} \in \Re^{I_1 \times I_2 \times I_3 \times \cdots I_N}$ of order N, having $\Re^n$ elements along each mode, the HOSVD is given by [15]:

$$\mathscr{X} = \mathscr{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \times_4 \mathbf{A}^{(4)} \ldots \times_N \mathbf{A}^{(N)}$$

Each element of $\mathscr{X}$ is thus calculated as:

$$x_{i_1 i_2 i_3 \cdots i_N} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g_{r_1 r_2 r_3 \ldots r_N} a^{(1)}_{i_1 r_1} a^{(2)}_{i_2 r_2} a^{(3)}_{i_3 r_3} \cdots a^{(N)}_{i_N r_N}$$

$$\text{for } i_n = 1, \ldots, I_n, n = 1, \ldots, N.$$

Note that the calculation for each element of $\mathscr{X}$ depends upon each element of $\mathscr{G}$. Although $\mathscr{G}$ is not diagonal, as with ***S***, if we sort the Frobenius norms of $\mathscr{G}$ in descending order, it is possible to obtain a similar ordering of the singular values as in **S**, which will become important later in interpreting the structure of generalized design knowledge.

The advantage of SVD and the HOSVD is their model for ***X*** or $\mathscr{X}$, respectively, permits the calculation of alternative representations of ***X*** or $\mathscr{X}$ through truncated forms of the singular vectors. In information retrieval [17] and the latent semantic approach [8], the truncated form is generally used to eliminate noise from the data, although in other work, the truncated form has also been shown to help in the re-formulation of complex design optimization problems [21], relax constraints in spatial topology planning problems [9], and identify community structure in complex networks [20]. The truncated vectors play a different role in characterizing generalized knowledge. The claim is that the orthonormal vectors in the truncated approximation of the original higher-order representation embodies generalized knowledge, because these orthonormal vectors are the most appropriate bases for the vectors in the original representation. Since these original vectors encode knowledge about designs, the span of the truncated orthonormal basis vectors must represent sufficient generalized knowledge about the designs so that the original knowledge (design representation) could be reproduced. Mathematically stated, they must represent a sufficient approximation so that the original matrix $\boldsymbol{X}$ or tensor $\mathscr{X}$ can be recalculated as a linear combination of the truncated orthonormal basis vectors/derivational modes.

The calculation of the truncated versions of the original matrix $\mathbf{X}$ or tensor $\mathscr{X}$ retains the k leading singular values of $\mathbf{X}$ or $\mathscr{X}$ as shown below:

$$\boldsymbol{X}_k = \boldsymbol{U}_k\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}}_k = \boldsymbol{u}_1\boldsymbol{s}_1\boldsymbol{v}_1^{\mathrm{T}} + \boldsymbol{u}_2\boldsymbol{s}_2\boldsymbol{v}_2^{\mathrm{T}} + \ldots + \boldsymbol{u}_k\boldsymbol{s}_k\boldsymbol{v}_k^{\mathrm{T}}$$

$$\mathscr{X}_{\mathrm{k}} = \mathscr{G}_{\mathrm{k}} \times_1 \mathbf{A}_{\mathrm{k}}^{(1)} \times_2 \mathbf{A}_{\mathrm{k}}^{(2)} \times_3 \mathbf{A}_{\mathrm{k}}^{(3)} \times_4 \mathbf{A}_{\mathrm{k}}^{(4)} \ldots \times_{\mathrm{N}} \mathbf{A}_{\mathrm{k}}^{(\mathrm{N})}$$

In the case of the matrix, the approximation is guaranteed to be optimal in the least squares sense, but in the case of the tensor, it can be optimal but not necessarily unique [7; 16]. However, the lack of uniqueness does not alter the basic premise that the truncation provides a generalization of the original knowledge, albeit with a mathematically different orthogonal basis.

In this section, we will study the characteristics or ‘signatures’ of three possible forms of generalized design knowledge: perfectly modular knowledge, hierarchically modular knowledge, and random knowledge, that is, knowledge with no regular structure. The concept of modularity captures the notion that designers generate knowledge by establishing connections between knowledge rather than relying on pre-established and fixed semantic categories [5] and the dynamic situatedness of design cognition [13]. Each of these modules of knowledge brings together objects that share relevant knowledge. In a perfectly modular knowledge structure, what is known about one aspect of a design neither affects nor influences knowledge about another aspect of the design. For example, to know how the LCD screen of a cell phone works, I do not also need to know how the Li-ion battery works, and knowledge of these two aspects of the cell phone are essentially independent. In a hierarchically modular knowledge structure, what is known about one aspect of a design also entails knowing about another aspect; in other words, knowledge about one aspect of the design is subsumed by knowledge of another aspect. Returning to the example of the cell phone, knowledge about a touch screen requires knowledge of both capacitive sensing and liquid crystal display technology. In between a perfectly modular and a perfectly hierarchically modular knowledge structure is a knowledge structure that has some modularity but some overlap as well, which is what we would expect for most real-world designs. Finally, the situation of a random knowledge structure, which we are unlikely to find in real-world examples, implies no known underlying structure for a design. This is impossible, given that any design by definition is 'planned' and therefore has some organization in its knowledge structure.

Let us start then with the simplest example of generalized knowledge structure that is perfectly modular. For illustration purposes, we model this knowledge structure empirically as a 2D network have 64 nodes (for example, 64 functions and 64 behaviors) and 8 modules (8 functions and 8 behaviors cluster together tightly), in other words, 64 units of knowledge and 8 groups of highly inter-related knowledge units with strong intra-module ties to each other but no ties outside of its module. Such a network might also represent a mapping between structural design parameters and functional requirements. Similarly, we can construct an order-3 tensor consisting of 64 units of knowledge along each of the 3 modes, representing for example Gero's FBS ontology of function, structure and behavior [12], or a functional modeling paradigm of product, function and flow [24], or the representation introduced earlier in terms of "stacks" of products from a single family with common attributes and functions. The modularity of the knowledge structure is obtained by taking a truncated approximation of the original knowledge representation and then clustering the nearest neighbors [20]. The darker blocks (in red) show a high level of interaction between knowledge units whereas the lighter colored blocks (in blue) show a weak or no interaction between knowledge units. There will always be a diagonal line of dark (red) blocks since each knowledge unit is related to itself. By using a value of k=16 in the truncation, Figure 2(a) reveals the modularity for a 2D knowledge representation and Figure 2(c) for 3D knowledge representation, for which the modularity in mode-1 and mode-2 of the tensor are identical and perfectly modular, and in mode-3 of the tensor, there is a regular knowledge structure, that is, partially modular.

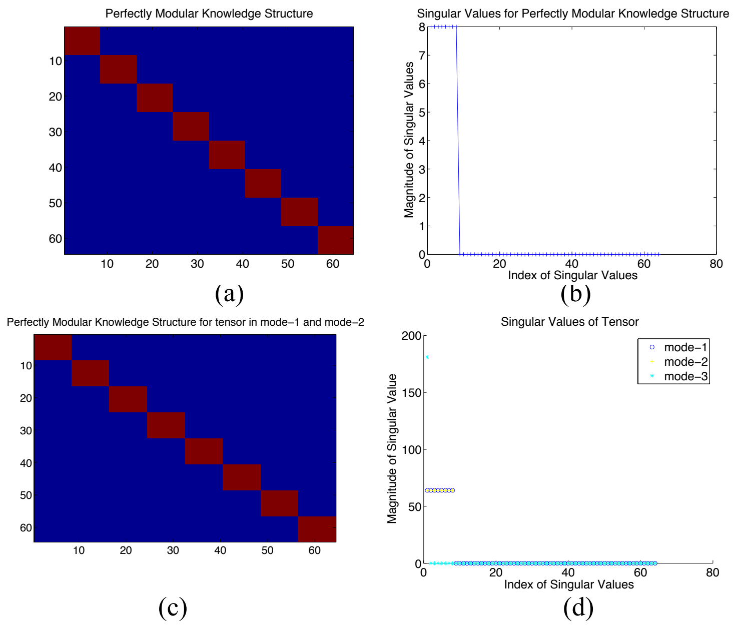

Fig2. Characteristics of Modular Knowledge Structure

We can note a few characteristics from the graphs. The first is that the number of non-zero singular values Figure 2(c) for the perfectly modular knowledge structure of Figure 2(a) is exactly the same as the number of modules. This holds true for the multidimensional knowledge structure of Figure 2(c) for mode-1 and mode-2. In mode-3, the knowledge structure (not shown) is a single block of knowledge because there is only one significant, non-zero singular value, wherein each unit of knowledge is strongly related to all the other units of knowledge. This is reflected in Figure 2(d), for which the number of significant non-zero singular values in mode-3 is 1; there is a long trail of singular values of magnitude less than 10^{-12}. In other words, the number of significant, non-zero singular values gives an indication as to the number of modular knowledge units in the knowledge structure and the degree of inter-relatedness of the knowledge.

We now turn to the case of a perfectly hierarchical modular knowledge structure. To generate the hierarchical modular structure, we first start with a perfectly modular structure and then introduce perturbations in this matrix using a parametric probability *p* of "rewiring", that is, introduce or remove a relation between two knowledge units. Multiple levels of rewiring, with a decreasing value of *p* at each stage of rewiring produce a matrix with a hierarchically modular structure. Figure 3(a) and Figure 3(c) show the hierarchical knowledge structure for a two-dimensional and three-dimensional knowledge structure, and Figure 3(b) and Figure 3(e) show the corresponding singular values. The signature of the singular values differs from the perfectly modular case; rather than a single discrete discontinuity between the significant non-zero singular values and the zero magnitude values, there is a series of discrete steps. The number of these discrete steps corresponds to the number of hierarchical levels, that is, the number of chunks of knowledge subsumed within larger chunks of knowledge. Figure 3(b) shows the SVD signature of this knowledge structure as having distinct gaps between the 2nd and 3rd, 4th and 5th, and 8th and 9th singular values, corresponding to three distinct hierarchical levels.

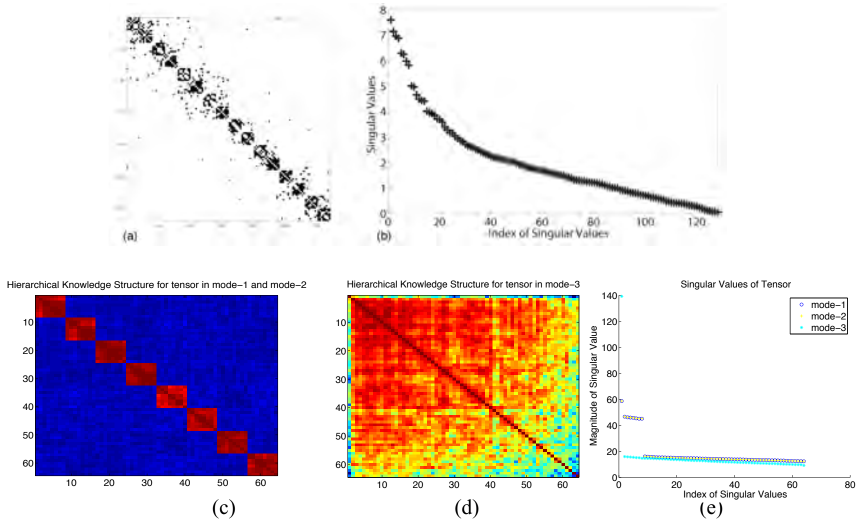

Fig3. Characteristics of Hierarchical Knowledge Structure

Finally, we turn to the situation of the random knowledge structure. A random knowledge structure is generated by connecting an edge between any two knowledge units if a randomly generated probability is greater than or equal to a probability p. There is neither a perceivable structure in the knowledge structure for the 2D case of Figure 4(a) nor mode-1 of the 3D case of Figure 4(b), which is equivalent to mode-2 and mode-3 (not shown). Corresponding to this, the singular values follow a trend wherein there is a single significant non-zero singular value followed by a ‘long tail’ of non-zero singular values. The magnitude of the first singular value is correlated to the degree of randomness; as p increases, the magnitude of the first singular value also increases, as shown in Figure 4(d).

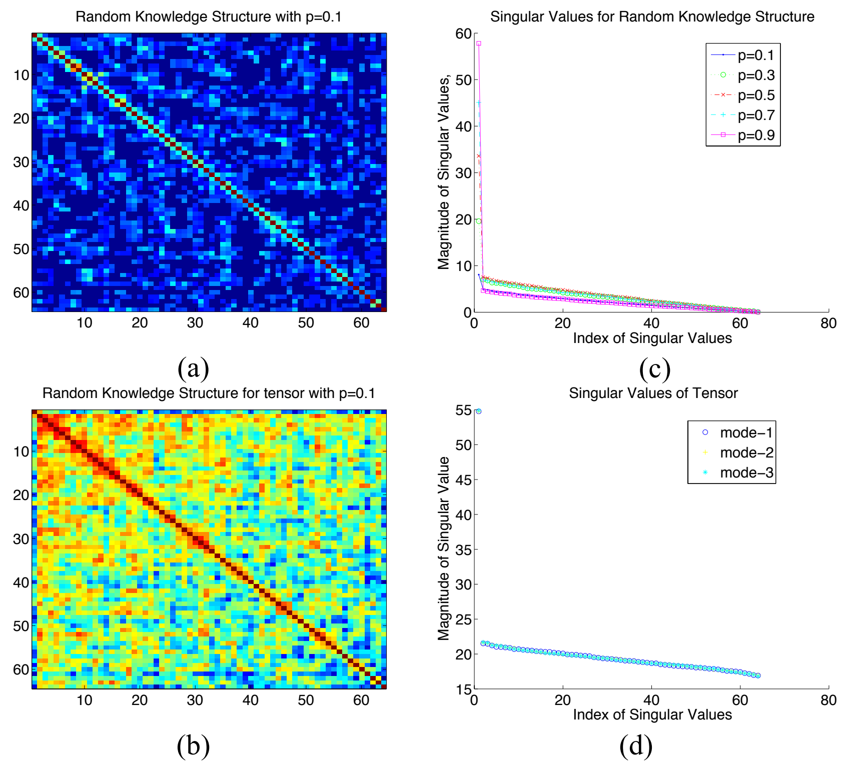

Fig4. Characteristics of Random Knowledge Structure

The preceding empirical results thus provide a way to characterize the underlying knowledge structure as being modular, hierarchically modular, or random. In particular, there are several characteristics that are particularly useful in identifying the type of knowledge structure, based on the singular value spectrum:

1. If the singular value spectrum shows discrete steps between clusters of singular values, and if the singular values are very similar within the cluster, then the knowledge structure is likely to be modular.
2. If the singular value spectrum shows discrete steps between clusters of singular values, and if the singular values vary linearly within the cluster, then the knowledge structure is likely to be hierarchically modular.
3. The number of significant singular values in a modular or hierarchically modular knowledge structure is related to the number of knowledge modules.

4. If the singular value spectrum is continuous with a single dominant non-zero singular value, then the knowledge structure is likely to be random.

With the above qualitative characteristics, we now show the underlying generalized knowledge structure of some real-world systems.

Knowledge Structure of Design Processes

In this section, we analyze the modularity structure of design processes. Though the preceding representations have been shown for representing products, the same matrix or graph representation is easily extended to the analysis of design tasks or processes. In our analysis we consider four product development datasets [2; 3] dealing with vehicle development, operating system software development, pharmaceutical facility development, and a 16-storey hospital development. The authors showed that these networks had a high degree of clustering, a finding that led them to conclude that the network structure of design and problem solving processes is modular in nature. We have obtained all datasets from the New England Complex Systems Institute [1]. In this data, a product development (PD) network is a directed graph with N nodes and E edges, where there is an edge from task v_i to task v_j when v_i feeds information to v_j. The 4 datasets are shown in Figure 5. Each of these networks record design process model diagrams, module-subsystem type design dependencies, structured interviews with designers and engineers in response to questions such as “from which tasks do the inputs to a particular design task come from and to what tasks does it feed?”, and various design documents. For example, in the pharmaceutical facility and hospital design projects, examples of tasks can be computations of grid layouts and computations of final pile layouts and spacings based on column positions [2; 3].

Using the SVD based method we propose here, we show that as the size of PD networks scale up, along with product size, their modularity *does not* scale up. We show that large size modules are relatively absent in very large PD networks, although they have many small sized modules. This is counterintuitive because the architecture of the product itself that is being designed using the corresponding process structure could be modular, with large sized modules present at the system level. It also brings out some important constraints that can govern the formation of knowledge structures in design processes, especially ones wherein time plays a critical role.

Figure 5 shows the detailed results for the largest dataset, the 16-storey hospital design project, with 889 tasks. The observations we present on this case, however, are also all valid for the other 3 datasets. First, these matrices are very sparse. The denser the matrix, the more the interaction time spent on design tasks. Thus, the reason for the sparseness is the objective of capturing the most amount of knowledge interaction in the least possible amount of tasks or activities.

Second, note that the singular values plot resembles a hierarchically modular network, with the singular values clustered in groups and large inter-group gaps, but the hierarchies and modules show a much smoother qualitative form as compared to the perfect hierarchical modular form presented earlier. While the first two singular values sit sharply separated from the others, there is a gradual reduction in the values for the others, with successive singular values just a little smaller than the previous ones. This characteristic is much more similar to a regular graph, which describes activities just next to each other as being connected the most, and the farther two activities are from each other, the lower the likelihood of connection between them.

Third, note that the rank of the matrix is 606, much less than the original size 889. This shows a large amount of redundancy in the data. At k=606, the full information is captured. Finally, note the reduced dimension representations, Figure 5(c)-(d). We have preserved the first 16 singular values and vectors, and we analyzed the modularity. It is immediately obvious that there are no large sized modules, and that all modules are much smaller as compared to the full matrix size.

These characteristics suggest that product development / design process networks differ from other social, technological and information networks of human interaction. In social networks, for example, it is well known that the module sizes scale with the network size, to an upper limit [10]. It is clearly visible that from a cognitive perspective, design and problem solving, or indeed any other social goal directed set of activities between humans involving exchange of information, have a distinctly different knowledge structure than networks of informal social interaction.

In the k=16 representation, a large part of the reordered network is black, or blank, signifying that a certain set of activities have no or negligible relationship with other tasks or activities. This shows that in product development networks, a major motivation is to have tasks and activities that are self-contained – their completion does not depend on interaction with many other activities, and hence they will play a significant role in speeding up the completion of the project and play an exemplary role in knowledge modularity. These tasks can be said to be perfectly modular – a single task with defined knowledge that can be completed without knowledge input from any other task.

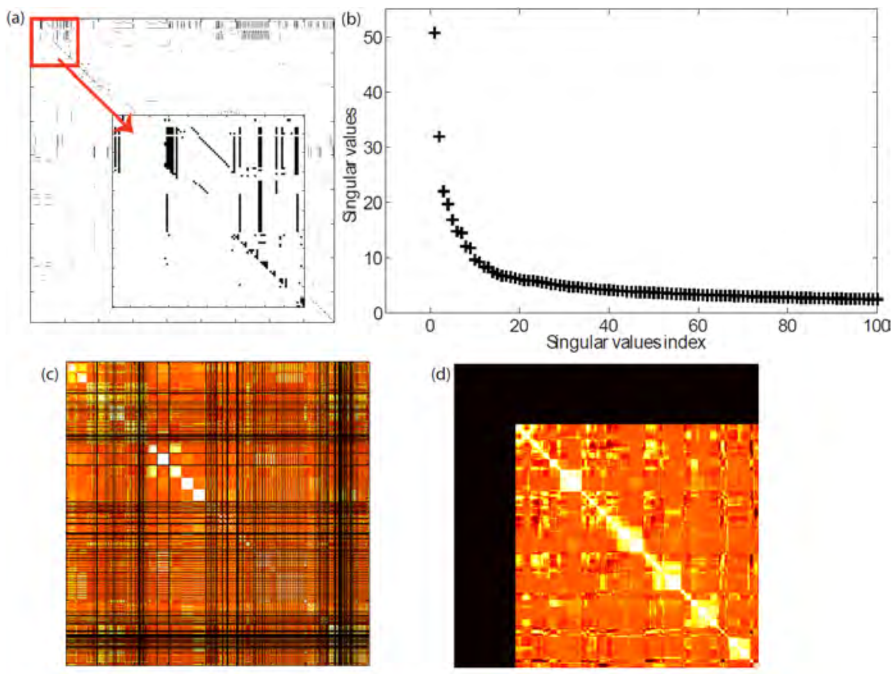

Fig5. SVD analysis for 16-story hospital design task knowledge matrix: (a) adjacency matrix, 889; a blow up of the top left hand corner of the matrix is shown for clarity (b) Singular values plot; (c) *k*=16 lower-dimensional representation; (d) reordered *k*=16 representation, showing modules

Based on above observations, we present two hypotheses. Our first hypothesis is that the presence of knowledge or task modularity in process networks is essential to ensure that sets of related tasks being completed need not be done or visited again. For example, the set of tasks involved in laying the foundation of a hospital building must be completely finished before the set of tasks dealing with constructing the higher storeys. This is the "module formation" hypothesis. Our second hypothesis is that large sized knowledge structures in process networks are absent or relatively rare, because they increase the iteration time to task completion within a single module. For example, in a software development module, the larger the size of the module, the more number of iterations needed to finalize it, and the lower its reusability as a generic component. However, given that the authors in the original paper extracted this information from personal interviews and design documentation, we could reasonably conjecture that the practitioners conceptualized their general knowledge of tasks in small chunks of constant size regardless of the size of the overall task. This constant size chunking may be caused by limited short-term memory when being asked to recall relations between tasks. This is the “module size limiting” hypothesis. In other words, we show that product development knowledge structures have the unique form they do because of two opposite formational forces – one forcing the formation of modules, the other putting constraints on the size of modules.

Finally, we note that the modules found in knowledge process networks at various hierarchical levels represent sets of design / management processes that are tightly coupled, and therefore, could be used to inform strategic knowledge in a design domain.

Knowledge Structure of Designed Objects (Products)

In the following example, a set of products was modeled using the functional basis [24]. The aim of this analysis is to compare the generalized knowledge structure identified by the HOSVD-based analysis and the common products and dominant functions and flows identified by Stone and colleagues. The tensor describing the products has a dimensionality of i=70 products, j=34 functions, and k=20 flows. A non-zero entry in the tensor means that product i implements function j on flow k. Consistent with the original data, entries in the tensor represent customer importance of the function-flow for the given product, and ranges from 0 (not important/not implemented) to 22. Where a product implements a function on a combination of flows, e.g., “*mix* liquid and solids” and “*mix* solids”, then the importance values were summed. Flows that did not appear in the data set even though they occur in the functional basis, such as “human motion” and “particle velocity”, were ignored. Sub-flows that did not appear in the data set were ignored and only the higher-level flow was included, e.g., “energy / electromagnetic / solar” but not “energy / electromagnetic / solar / intensity”. Some minor changes were made to the data set, such as setting the function-flow for the Krups Café Trio to “measure hydraulic pressure”, since the device pressurizes steam to froth the milk, and the Paint Sprayer to “measure pneumatic pressure”, since the device operates on pressurized air to spray the paint out the nozzle. Both devices were recorded in the data set as “measure pressure”, which was ambiguous. In the data set provided, the verb “dissipate” was used as a function. However, this verb does not appear in the functional basis vocabulary. The function dissipate was added to the category “Control Magnitude / Change” since energy dissipation involves controlling the loss of energy in an orderly manner. Figure 6(a), (b), and (c) show the non-truncated knowledge structure for the data set, in which reveals no apparent knowledge structure.

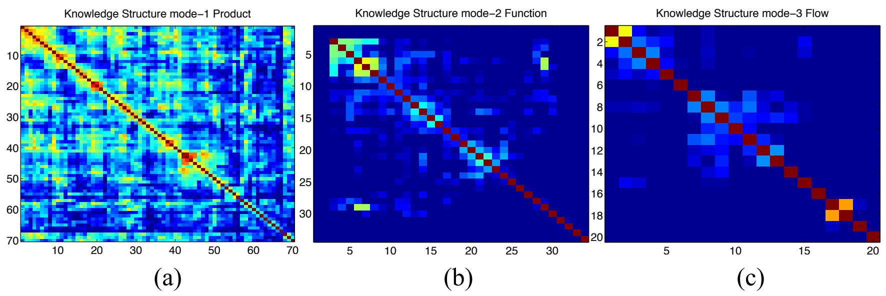

Fig6. Non-truncated knowledge structure for products in functional basis

To reveal the underlying knowledge structure, a truncated form of the original knowledge structure is generated by retaining the leading singular values in each mode of k_1=12, k_2=12, and k_3=13. These values were empirically obtained by selecting the index of the singular values when the magnitude of the singular value dropped below 20, at which the magnitudes for the singular value for each mode crossed over and dropped significantly. The knowledge structures were then generated, and shown in Figure 7(a), (b), and (c), for product, function, and flow, respectively.

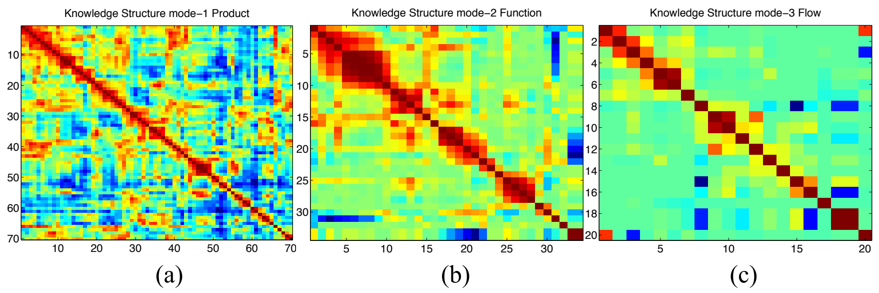

Fig7. Knowledge structure for products in the functional basis

Some common knowledge structures, i.e., products that appear in the same family of products, dominant functions, or dominant flows, which are consistent with the product module architectures identified by Stone and colleagues [25], include:

- Coffee-maker family of products, appearing as a small module in the middle of Figure 7(a), consisting of the items {'West Bend Iced Tea Maker-RBS', 'Mr. Coffee Coffee Maker-RBS', 'KRUPS Café Trio', 'Mr. Coffee Iced Tea Maker-RBS'}

- Functional module associated with providing information, consisting of {‘Signal/Indicate’, ’Signal/Measure, ’Signal/Display’}
- Human hand flows, consisting of the flows {‘Material/Human’, ’Energy/Human/Force’}, which are the only human flows appearing with more than one function. The human flow of Human energy alone appears only with the function Import.

In addition, there are overlaps in knowledge structure, implying the ability to transfer the knowledge. For example, the following products share an overlap with at two other product modules: {‘B&D Weed Trimmer’, ‘Durabuilt Hand Vacuum’, ’Wet/Dry Vacuum’, ’SKIL Power Screwdriver-RBS’}, which means that the ‘electricity to torque’ module of the SKIL Power Screwdriver and the ‘electricity to pneumatic energy’ of the vacuums could be shared with the Hamilton Electric Mixer and the Pneumatic Air Ratchet. An important overlap identified by Stone and colleagues are the modules supplying electricity and driving “resulting in a mix modularity tool” [24, p. 17], which is explicitly identified in the module {‘Connect / Mix’, ’Provide / Store’, ’Convert’}. This modular overlaps with the module {‘Provide / Store’, ’Convert’, ’Control Magnitude / Change / Condition’, ’Guide / Translate’, ’Control Magnitude / Change / Form} and is hierarchically embedded within the larger module {‘ Separate’, ’Connect / Couple’, ’Transfer / Transport’, ’Guide / Allow DOF’, ’Support / Stop’, ’Refine’, ’Distribute’, ’Connect / Mix’, ’Provide / Store’, ’Convert’, ’Control Magnitude /Change / Condition’}.

Conclusions

This paper proposed a way to describe the underlying generalized knowledge structure of designed objects and design processes. The approach is valid for two-dimensional and multi-dimensional product and process representations. The purpose is to find an approach to obtain broad knowledge frameworks for a set of design representations that may not appear to share any *a priori* similarity.

The results on synthetic and empirical data sets demonstrate the feasibility of the approach in generating plausible knowledge structures. As the results suggest, knowledge structure discovery over a broad base of design representations provides a way of acquiring experience-driven constraints on the structure of design concepts. Depending upon the type of experiences (or design representations) presented, different concepts can emerge across explicit and implicit relations, and with each subsequent experience having the potential to displace prior concepts. Different from machine learning systems used to find patterns in design representations based on classifying pre-specified properties, the approach operates on the premise of finding structure in any representation of similarity *or* relations between design parameters or functional requirements of designed products or design processes. Finding these broad knowledge frameworks from experience is crucial for transferring design knowledge across domains, wherein explicit relations are rarely clear or defined, but often discovered.

References

1. Braha D, Bar-Yam Y: 2000, Data for The Topology of Large-Scale Engineering Problem-Solving Networks http://necsi.edu/projects/braha/largescaleengineering.html 02 January 2011.
2. Braha D, Bar-Yam Y (2004) Topology of large-scale engineering problem-solving networks, Physical Review E 69(1): 016113
3. Braha D, Bar-Yam Y (2007) The Statistical Mechanics of Complex Product Development: Empirical and Analytical Results, Management Science 53(7): 1127-1145
4. Browning TR (2001) Applying the design structure matrix to system decomposition and integration problems: a review and new directions, Engineering Management, IEEE Transactions on 48(3): 292-306
5. Coyne RD, Newton S, Sudweeks F (1993) A Connectionist View of Creative Design Reasoning, in JS Gero ML Maher (eds), Modeling creativity and knowledge-based creative design, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 177-209
6. Darke J (1979) The primary generator and the design process, Design Studies 1(1): 36-44
7. de Lathauwer L, de Moor B, Vandewalle J (2000) A multilinear singular value decomposition, SIAM Journal on Matrix Analysis and Applications 21(4): 1253-1278
8. Dong A (2005) The latent semantic approach to studying design team communication, Design Studies 26(5): 445-461
9. Dong A, Sarkar S (2011) Unfixing design fixation: from cause to computer simulation, Journal of Creative Behavior 45(2): 147-159
10. Fortunato S, Barthélemy M (2007) Resolution limit in community detection, Proceedings of the National Academy of Sciences 104(1): 36-41
11. Gero JS (1990) Design prototypes: a knowledge representation schema for design, AI Magazine 11(4): 26-36
12. Gero JS (1998) Concept formation in design, Knowledge-based systems 11(7-8): 429-435
13. Gero JS, Kannengiesser U (2004) The situated function-behaviour-structure framework, Design Studies 25(4): 373-391

14. Kemp C, Tenenbaum JB (2008) The discovery of structural form, Proceedings of the National Academy of Sciences 105(31): 10687-10692
15. Kolda T (2009) Tensor Decompositions and Applications, SIAM Rev. 51(3): 455
16. Kroonenberg P, de Leeuw J (1980) Principal component analysis of three-mode data by means of alternating least squares algorithms, Psychometrika 45(1): 69-97
17. Landauer TK, Foltz PW, Laham D (1998) Introduction to Latent Semantic Analysis, Discourse Processes 25: 259-284
18. Leslie AM (1987) Pretense and Representation: The Origins of "Theory of Mind", Psychological Review 94(4): 412-426
19. Perner J (1991) Understanding the Representational Mind, The MIT Press, Cambridge
20. Sarkar S, Dong A (2011) Community detection in graphs using singular value decomposition, Physical Review E 83(4):
21. Sarkar S, Dong A, Gero JS (2009) Design optimization problem (re)-formulation using singular value decomposition, Journal of Mechanical Design 131(8):
22. Schön DA (1963) Displacement of concepts, Tavistock Publications, London
23. Sinclair G, Klepper S, Cohen W (2000) What's Experience Got to Do With It? Sources of Cost Reduction in a Large Specialty Chemicals Producer, Management Science 46(1): 28-45
24. Stone RB, Wood KL (2000) Development of a Functional Basis for Design, Journal of Mechanical Design 122(4): 359-370
25. Stone RB, Wood KL, Crawford RH (2000) A heuristic method for identifying modules for product architectures, Design Studies 21(1): 5-31
26. Strang G (1988) Linear algebra and its applications, Harcourt, Brace, Jovanovich, Publishers, San Diego
27. Summers JD, Bettig B, Shah JJ (2004) The Design Exemplar: A New Data Structure for Embodiment Design Automation, Journal of Mechanical Design 126(5): 775-787
28. Tucker LR (1964) The Extension of Factor Analysis to Three-Dimensional Matrices, in N Frederiksen H Gulliksen (eds), Contributions to Mathematical Psychology, Holt, Rinehart and Winston, Inc., New York, 110-127
29. Ward TB (1994) Structured Imagination: the Role of Category Structure in Exemplar Generation, Cognitive Psychology 27(1): 1-40