A Problem Decomposition Method for Conceptual Design

Somwrita Sarkar, Andy Dong

Design Lab, Faculty of Architecture, Design and Planning,
University of Sydney, Australia
email: ssar3264@mail.usyd.edu.au, andy@arch.usyd.edu.au

John S. Gero

Krasnow Institute for Advanced Study, Virginia, USA and UTS,
Sydney, Australia
email: john@johngero.com

Abstract: In this paper, based on findings from situated cognition, we argue that human reasoning is characterized by an ability to dynamically re-organize knowledge available in an experience, and re-construct it for use within the same experience. This paper describes a computational method that models such a knowledge re-organization process. We show its application on the Design Structure Matrix (DSM) formulation of an automotive design problem to demonstrate a design decomposition strategy. The method infers decomposition decisions by uncovering information from syntactic representations of design problems, i.e. it reformulates the problem by a purely syntactical analysis of a formulation. In general, it infers global, implicit design knowledge from local, explicit information in the design representation, and shows that a single problem representation can encode multiple layers of 'design meaning'. Designers can use the method to explore problem decomposition decisions for large/complex systems in an interactive manner.

Keywords: Design theory and methodology, Situated cognition, Design decomposition, Conceptual design, Dynamic design support tools, DSM, SVD

1 Motivation and Significance

Decomposition of complex systems and products is an important part of formulation performed at the systems design stage. Decomposition decisions taken at the pre-modeling stage can significantly affect the way in which design problems as components within a larger system are modeled. "Complex" product design is typically characterized by either large problem sizes in terms of the number of design variables and constraints, or strong interactions between them [1], usually defined by analytical functional constraints [2] or logical dependencies between variables [3]. Whether the complexity stems from a large problem size or strong interactions, solving such models for a design result is difficult and frequently involves the application of numerical algorithms onto the problem model. Large model sizes reduce the speed and efficiency of solution algorithms [2]. Design decomposition is a common approach for reducing problem complexity to ensure ease and speed in finding a solution. It is a process that breaks down a large

problem into smaller sub-problems with local variables and constraints with the parent problem defined through linking variables and constraints. The smaller sub-problems may then be solved independently of the main problem, thereby simplifying the solution procedure.

Efficient reduction of complexity occurs when the decomposition strategy leads to "nearly decomposable systems" [4] characterized by weakly interacting sub-systems with strong local interactions within each sub-system. The final aim of the process is to obtain a decomposition that minimizes the effort to solve a problem by, for example, maximizing the modularity of the system. Obtaining a "good" decision is essential, as system configurations once taken for a certain product are changed generally at high cost once such decomposition decisions are made. The "optimal" decision is agreed upon using trial and error, and is usually too expensive to alter in terms of the process, team, material or equipment changes required [3].

Design decomposition algorithms have received considerable attention in the research literature and are mainly classified into object-based, aspect-based or sequential [2]. Despite a wide range of computational algorithms being available for this task, final decisions for most complex systems are still taken manually, mainly because of the subjectivity involved in decision making due to the range of criteria involved. Significant trial and error is required to develop decomposition decisions in systems design before and while a detailed mathematical model with analytical constraints or formal relationships is finalized. Therefore, in addition to "exact" decomposition algorithms, it would be beneficial to develop "exploratory" design methods that assist in this trial and error by allowing the designer to quickly redefine problems and observe changed results. Such a method should have two important capabilities – (1) if a change is made to one part of the system (for example, changing the design interaction or dependency between two elements in a Design Structure Matrix representation), then this should reflect in the way the whole system is affected (for example, the coupling measurements between the various sub-systems in the main system); and (2) it should be able to 'see' multiple potential decisions, as frequently, for complex design problems there is no one dominant solution, but many feasible or optimal solutions.

In this paper, based on findings from the field of situated cognition, we argue that human reasoning is characterized by an ability to dynamically re-organize knowledge available in an experience, and re-construct it in multiple ways for use within the same experience. This allows for 'exploratory' behavior, an important ability is missing in most computational design tools or design automation algorithms. This paper describes a computational method that models such a knowledge re-organization process using an unsupervised pattern extraction perspective. The method has been developed as a general mechanism for automating several aspects of design problem (re-)formulation, and we have previously applied it to formal design optimization problems [5, 6]. In this paper, we show its application to the Design Structure Matrix (DSM) formulation of a design problem to demonstrate an efficient decomposition strategy. The strategy infers decomposition decisions from syntactic representations of design problems. It infers design information from the existing syntax of a problem formulation to reformulate the same problem. Further, if a local explicit relationship between any two variables/system components is changed by a designer in the original problem formulation, the method can show the global effect this has on the whole system, with potential changes to the decomposition decisions. That is, the method allows a designer to change the problem formulation, and query for multiple decomposition results in an interactive manner. The

DSM representation is chosen because it only specifies dependencies between design elements, specially suited for design problems in which analytical relationships between variables may not be available or are too expensive to model when conceptual decisions are being taken for systems design.

Such a method will be useful for providing design support for the decomposition of complex systems and products. Designers can use this method to input an initial formulation and query the system for decomposition decisions. That is, the method returns a reformulated problem as an answer to the initial query formulation. If the current decision is not acceptable, or any new requirements and relationships are to be modeled between design variables, they can change or add new information and reapply the method to see how the results change, in an interactive manner. As systems and products grow more complex, designers will benefit from design support tools that can incorporate such robust computational mechanisms to assist with cognitive tasks traditionally performed by designers.

2 Situated Cognition and Conceptual Design

The theory of situated cognition [7] grew out of a critical analysis of the differences in the way human knowledge is represented in the mind as opposed to inside a computer program. Situated cognition says that knowledge is dynamic, and results from a first person, subjective interaction with the world [8]. In this sense, all knowledge is "at least partially improvisatory" [7] and therefore, always like a dialogue in construction. This dialogue occurs between an agent (e.g. artist, designer, athlete, scientist etc.) and the task (e.g. painting, music composition and improvisatory performance, constructing design representations or "doing design", running, scientific theorizing etc.). The course of the task is directed dynamically by the "dialogue". Thus, some knowledge gained from the experience of performing a particular task will affect the actions that are taken next. For example, if a designer performs design decomposition for a certain system, the results of this decomposition process will be used by the designer to further change the problem representation and redo the decomposition in an iterative manner. In any experience, parts of objective knowledge will be recalled, but parts have to be subjectively re-constructed as a result of what is currently happening in the design process. This is the fundamental ability that humans demonstrate effectively – an ability to dynamically re-organize knowledge available in an experience, and re-construct it for use within the same experience. Designing has been defined to be a situated activity [8] and the resulting design is a situated product. The question of how to computationally model tools that can interact with the design task, and support the designer adaptively and robustly in a design task, *while the design task is in progress* has been a central question in design computation. In this paper, we present a computational method that is able to do this in the domain of mathematical-symbolic design representations dealing with design decomposition tasks in complex systems design.

3 The Method

The method we present here has been developed as a general mechanism for automating many tasks in symbolic design problem modeling and reformulation [5, 6]. One main

advantage of the method is that the same mathematical process may be used to perform a variety of problem reformulation tasks. Examples include design decomposition (that we demonstrate in this paper), heuristic design "case" identification, modularity analysis, identification of linked variables and functions, topology modeling (these are demonstrated elsewhere and not in this paper). Further, the method can be applied to both analytically and non-analytically stated problem models. The method takes a standard mathematical problem model (such as an analytical design optimization model or a DSM model) as input, and converts this into an occurrence matrix that captures information on mutual relationships of design elements. A Singular Value Decomposition (SVD) is then performed over this matrix followed by dimensionality reduction, similarity measurements and unsupervised clustering. Figure 1 shows the various stages of the method.

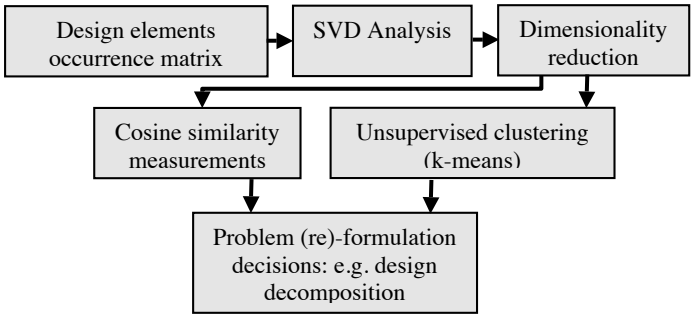

Figure 1 The inference algorithm for problem modeling and (re-)formulation decisions

SVD is a linear algebra based factorization method in which any a rectangular matrix **A** is factorized into three matrices $\mathbf{A} = \mathbf{USV}^T$, where **U** and **V** are orthogonal matrices and **S** is a diagonal matrix of singular values. The conceptual idea behind the mathematics is that any matrix **A** in which the rows and columns stand for design concepts (variables, parameters, constraints, etc.) can be factorized into independent components represented by a set of new, abstract variables that contain information about how all original entities are correlated. The decomposition measures, through linear combinations, the association or coupling strength of each design concept's relationship with all the others as implied by the matrix **A**. In other words, it uses local explicit information in the matrix to produce global implicit knowledge inherent in the design formulation. The special behavior of SVD is that even if one local matrix element is changed in **A** (implying a single design dependency between two design elements in a DSM), this will reflect as a global change as all the $\mathbf{USV}^T$ values will change. Further, if such a factorization is then approximated to a lower dimension space (in a linear least squares sense), then it is possible to extract implicit design relationships that cannot be observed directly from the explicit relationships in the matrix. The method is able to capture the coupling strength between two design elements that may not be explicitly related in the DSM because it captures associative patterns between all elements and re-represents them in a common representation space. The implication is that co-occurrences of symbols in design representations embed implicit "meaning" about the artifact that is not explicitly evident from the representation itself. Because the SVD algorithm scales very well, the method is applicable from very small to very large problems. For a detailed description of the method, refer to [5,6].

The idea of performing SVD over a data matrix followed by dimensionality reduction is not new. It is, for example, routinely used in the statistical natural language processing [9] and digital image processing disciplines [10] to reveal semantic patterns in the data in an unsupervised manner. What is interesting is that it reveals semantic patterns from syntactic data in such apparently diverse knowledge domains. We developed a similar methodology for design domains with the hypothesis that it would be able to reveal the implicit and explicit semantics of design knowledge from the syntax of design representation. In our previous work, the rows stood for design variables and the columns stand for analytical functional constraints and the method was applied and tested on design optimization problems. In this paper, the rows and the columns stand for the same design elements, and we apply and test the method with the aim of performing design decomposition. We test the method in such a situation where the only information available is the possible interactions that might occur between the various design elements of the system. Because of such a problem form, the Design Structure Matrix (DSM) formulation seemed ideal.

4 Application and results: automotive climate control system design

Figure 2 shows the DSM representation for a Ford automotive climate control system [3]. The system is to be decomposed component-wise into sub-systems with four kinds of constraints (interaction types): spatial adjacency, energy interactions, materials interaction and information exchange between components. The interaction strengths range from -2 to +2, signifying negative as well as positive interactions. For example, because the radiator and engine fan are functionally coupled with each other, they get a +2 score for both spatial adjacency and material interaction conditions.

In the DSM representation, the diagonal matrix entries (the relationship of a design element with its own self) are zero. In the occurrence matrix form used for our method, we assume the maximum value +2 in the diagonal with the assumption that any design component has a maximum interaction with itself in terms of all the four interaction types considered. Each of the four interaction types (spatial, energy, information and material) impose different sets of constraints onto the design components and may result in conflicting decisions on how the components are to be decomposed as sub-groups. Also, these are likely to lead to different design team formation decisions specializing in different design aspects.

The authors in the original paper [3] do not describe in detail the clustering algorithm they have employed in reaching the results. They state that they have used a heuristic swapping algorithm that operates on the basic principle of re-ordering rows and columns such that the positive elements cluster round the main diagonal. In our method, we achieve the same results using a different approach. In addition, due to the global method structure (as opposed to the local method structure employed by the authors of the source paper), our method is able to provide insight into multiple possible reformulation decisions in terms of varying coupling strengths between design elements viewed at different levels of abstraction. SVD analysis factorizes the original matrix, re-representing the design elements as vectors in a space. Distances between any two design elements show the strength of their coupling as per the explicit relationship information in the occurrence matrix. Dimensionality reduction approximates the original matrix to a lower dimensional space. In this space, it is now possible to observe implicit associative

semantic relationships between all elements, even those that show no direct relationship in the original data matrix. For example, suppose that elements (A and B) and (B and C) have high positive relationships in the original formulation, but (A and C) have a zero relationship. The explicit local relationship between A and C is thus a low coupling strength. In the dimensionality reduction step, the implicit relationship that is retrieved will be that A and C will show a high positive relationship on account of the fact that they share a positive relationship with the same common element B. Cosine measurements between these vectors measure the distance of one design element from the other, i.e. how "close" or "distant" the design elements lie from each other in terms of the particular interaction type being considered in this re-represented space. Cosine measurements provide the opportunity to cluster values. Mutually high cosine values between elements imply high coupling strength suggesting they should be clustered together as a sub-problem. A cosine threshold value decides which elements belong to which cluster. Applying the k-means algorithm provides a parallel and similar analysis tool. Equivalent results are produced by cosine angle analysis and k-means clustering.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Radiator A		2 0 0 2			2 -2 0 0											
Engine Fan B	2 0 0 2				2 0 0 2								1 0 0 0			
Heater Core C				1 0 0 0			2 0 0 0	-1 0 0 0								0 0 0 2
Heater Hoses D			1 0 0 0						-1 0 0 0							
Condenser E	2 -2 0 0	2 0 0 2				0 2 0 2		-2 2 0 2								
Compressor F					0 2 0 2			0 2 0 2	1 0 0 2	0 0 2 0	0 0 2 0		1 0 0 0			
Evaporator Case G			2 0 0 0					2 0 0 0						2 0 0 0	2 0 0 0	2 0 0 2
Evaporator Core H			-1 0 0 0		-2 2 0 2	0 2 0 2	2 0 0 0		1 0 0 2							0 0 0 2
Accumulator I				-1 0 0 0		1 0 0 2		1 0 0 2		1 0 0 0						
Refrigeration Controls J						0 0 2 0			1 0 0 0		0 0 2 0		1 0 0 0			
Air Controls K						0 0 2 0				0 0 2 0		0 0 2 0	1 0 0 0	0 0 2 0	0 0 2 0	
Sensors L											0 0 2 0		1 0 0 0			
Command Distribution M		1 0 0 0				1 0 0 0				1 0 0 0	1 0 0 0	1 0 0 0		1 0 0 0	1 0 0 0	1 0 0 0
Actuators N							2 0 0 0				0 0 2 0		1 0 0 0			
Blower Controller O							2 0 0 0				0 0 2 0		1 0 0 0			2 0 0 2
Blower Motor P			0 0 0 2				2 0 0 2	0 0 0 2					1 0 0 0		2 0 0 2	

NOTE: BLANK MATRIX ELEMENTS INDICATE NO INTERACTION (FOUR ZERO SCORES)

Legend:

Spatial: S E :Energy
Information: I M :Materials

Figure 2 DSM representation for automotive climate control system interactions: [3]

4.1 Decomposition results from the "Material" interaction type matrix

We applied the method on the "Material exchange" occurrence matrix. The original data matrix has 16 design elements. The occurrence matrix is thus a 16 X 16 size matrix. We demonstrate how preserving just the first 2 singular values in the dimensionality reduction step, i.e. re-representing the original data in to 2 dimensions, can produce all the relevant decomposed clusters. Figure 3 shows how the design elements are represented in a 2-dimensional space after performing the dimensionality reduction step, and how they are clustered by performing cosine similarity measurements and applying the similarity threshold.

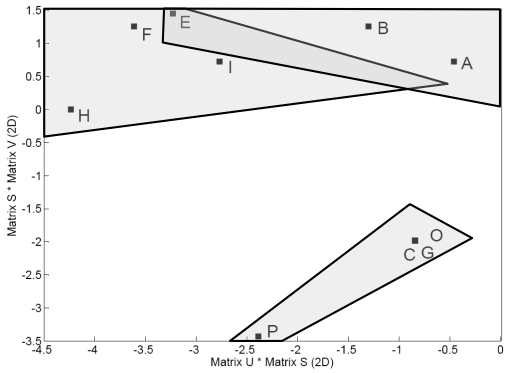

Figure 3 Design problem data in 2 dimensions and decomposition into sub-problems

Table 1 shows the cosine angle measurements between each of the elements in the 2D space (elements that do not participate in the material interaction type have not been represented). Elements within a cluster have high cosine measurements between themselves and low cosine measurements with elements from the other clusters, reaching a system configuration that follows the principle of a "nearly decomposable system" – weakly interacting sub-systems with strong interactions within each. To identify a cluster, a cosine threshold has to be used. All elements higher than this threshold are part of a cluster.

Table 1 Cosine measurements between design elements: interaction type MATERIAL; cosine threshold for clustering = 0.8

	A	B	E	F	I	H	C	G	O	P
A	1									
B	0.972	1								
E	0.834	0.941	1							
F	0.783	0.908	0.997	1						
I	0.732	0.872	0.986	0.997	1					
H	0.535	0.719	0.912	0.945	0.967	1				
C	-0.568	-0.358	-0.020	0.067	0.145	0.391	1			
G	-0.568	-0.358	-0.020	0.067	0.145	0.391	1	1		
O	-0.568	-0.358	-0.020	0.067	0.145	0.391	1	1	1	
P	-0.387	-0.159	0.184	0.270	0.344	0.571	0.979	0.979	0.979	1

Our method identifies three main clusters or chunks as a result of decomposition, considering 0.8 as the cosine threshold for membership within a particular cluster: (1) components A, B and E; (2) components E, F, I and H; and (3) components C, P, O and G. These results are almost identical to those reported in the original paper [3] with small differences. In our method, component H is classified only in the second chunk, instead of being a shared part of the second and third chunks as reported in [3]. This happens because H co-occurs (has positive interactions with) only component P in the last chunk and with no other element, thereby lowering its inclusiveness in the last chunk. Further, if

we lower the cosine threshold to 0.7, an alternate clustering is suggested: (1) components A, B, E, F, I, H and (2) components C, P, O, G. This is because components A and B show high cosine measurements with components F, I, H due to the fact that both these sets co-occur with E. Since the method works by measuring distributed patterns of associations between all elements it reveals implicit, indirect associations as well as explicit, direct ones. Even those elements that do not directly co-occur with each other show high semantic relationship with each other on the ground that they are related through co-occurrence with a third element. This is an interesting result because it shows that designers can experiment with varying the cosine thresholds to observe multiple ways in which the decomposition may be performed. This is an advantage, as frequently, there is no one right answer (dominant/optimal solution) for complex design problems and sub-system boundaries are subjectively decided. Designers need to "converse with" the design problem, i.e. interact with it from different perspectives, to be able to take decisions. This is one of the major strengths of this method – it can extract multiple decompositions from the same problem representation using the different cosine thresholds. This supports "exploratory" behavior.

Another possibility for observing multiple viewpoints and decisions [5, 6] is to vary the number of dimensions used in the dimensionality reduction step. Lastly, because the method captures implicit and explicit relationships between design elements in a distributed way, changing just one interaction value will produce changes in all the resulting matrices and cosine values, which may then alter the decomposition decision suggested. This allows designers to experiment with the matrix entries, i.e. the interaction values in an efficient way to observe how the decomposition decision may change.

Thus, it was shown that the method demonstrates an ability to dynamically re-organize the knowledge available in a problem representation (design experience), and re-construct it in different ways for use within the same experience (observing multiple decomposition decisions using implicit relationships between design components).

4.2 Decomposition results from the "Spatial" interaction type matrix

A similar analysis is done on the spatial interaction data matrix. The results (matching with the results reported in the original source [11]) show 3 tight clusters: (1) components A, B and E; (2) components P, O, N and G; and (3) components G, C and H. Table 2 shows the cosine similarity measurements.

The results show one minor difference from the original reported in the source [11] – note that in the C-H-G cluster, the cosine between components H and C is 0.419 (low), although in this C-H-G group all the other values are high thereby justifying its identity as a sub-problem. Referring back to the interaction data matrix (Fig. 2), we can see that there is an undesirable spatial relationship with value -1 between components H (Evaporator core) and C (Heater core). Thus, although spatial adjacency between G and H and G and C is highly desirable, leading to the decision of the H-C-G cluster, there is a conflicting undesirable spatial relationship within H and C in this cluster, and hence the low cosine value. This shows that when the clusters are being identified, if most elements within that cluster show high cosine measurements with a very few showing low values, this is a cue that there might be conflicting design information as input. Then, the designer can either alter input data to observe the changes in decomposition, or if this is not possible, then the designer is at least made aware that there is a conflict. This is relevant for large problems as it might not be possible to observe this manually from the explicit local data.

Table 2 Cosine measurements between design elements: interaction type SPATIAL; cosine threshold for clustering = 0.8; tables show three identified sub-problems

	A	B	E
A	1		
B	0.997	1	
E	0.982	0.994	1

	P	O	N	G
P	1			
O	1	1		
N	0.999	0.999	1	
G	0.986	0.986	0.989	1

	C	H	G
C	1		
H	0.419	1	
G	0.946	1	1

4.3 In summary

Decomposing the system from two perspectives suggests different clusters as sub-systems. The results largely match the decomposition results shown the source paper with minor differences. The decomposition offers the following insights: (a) components A, B and E are identified as a sub-system in both cases, thereby reinforcing their identification as a sub-system; (b) the other components show overlaps, and therefore, there is no single way to define clusters. The method can use the relationships in each different perspective to identify independent clusters, but the importance of each perspective has to be decided by the designer. For example, the spatial perspective may be defined as more important than the material [3] and the decomposition suggested by the spatial perspective can be given priority. The authors in the original paper note that using independent perspectives to build the product architecture and organize product development teams is more advantageous than taking an approach where a weighted combination of the criteria is used to superimpose individual results for a final one. Therefore, the individual perspectives are left as such with the insights that they provide, and they are not superimposed to identify final sub-system clusters.

5 Conclusions

We have presented a method for design decomposition in complex systems design that demonstrates an ability to dynamically re-organize the knowledge available in a problem representation (design experience), and re-construct it in different ways for use within the same experience (observing different decomposition decisions using implicit relationships between design components). The method infers global implicit design knowledge from the explicit local information in the problem formulation. The method has been developed as a general mechanism for supporting various mathematical-symbolic problem modeling and reformulation tasks in design. It provides an interactive exploratory platform to the designer to quickly redefine the problem and observe multiple problem reformulation decisions. One main advantage of the method is that the same mathematical process may be used to perform a variety of problem reformulation tasks. This paper demonstrates the way it is used to perform design decomposition, but we have used it to perform a range of other reformulation tasks. It is also a general method, because it can be used on analytically as well as non-analytically formulated design problems by converting them into a common representation form of the occurrence matrix.

For design decomposition, its strength, in comparison with other exact methods, is that it uses a single problem representation to uncover multiple encoded implicit

meanings, leading to multiple decisions. While most decomposition methods work with absolute coupling strengths (for example, with values such as +2, -2, 1, 0 etc. the authors of the source paper use a heuristic swapping algorithm to reorder rows and columns), this method turns these values into a range of continuously varying coupling strengths between components viewed at different levels of abstraction (cosine thresholds, and number of singular values or dimensions retained). Further, local, explicit changes made to one part of the system can show how the global relationships between all components change. These abilities allow this method to uncover multiple patterns, any of which could be chosen as a reformulation decision. These abilities make the method more robust in acting as a design support tool, because for complex design problems, there is frequently no "one" right decision or dominant solution and multiple solutions need to be considered.

Acknowledgments

This research is supported by a University of Sydney, Faculty of Architecture, Design and Planning International Research Scholarship.

References

1 Allison, J.T., Kokkolaras, M. and Papalambros, P.Y., "On selecting single-level formulations for complex system design optimization", Journal of Mechanical Design, 129, 898-906, 2007.

2 Michelena, N.F. and Papalambros, P.Y., "A hypergraph framework for optimal model based decomposition of design problems", Computational Optimization and Applications, 8(2), 173-196, 1997.

3 Pimmler, T.U. and Eppinger, S.D., "Integration analysis of product decompositions", Proceedings of the ASME Sixth International Conference on Design Theory and Methodology, Minneapolis, 1994.

4 Simon, H.A., "The sciences of the artificial", MIT Press, 1996.

5 Sarkar, S., Dong, A. and Gero, J.S., "Learning symbolic formulations in design optimization", in Gero, J.S. and Goel, A.K. (eds), Design Computing and Cognition 08, Springer, pp. 533-552, 2008.

6 Sarkar, S., Dong, A. and Gero, J.S., "A learning and inference mechanism for design optimization problem (re)-formulation using singular value decomposition", Proceedings of ASME Design Theory and Methodology Conference, New York, 2008 (to appear).

7 Clancey, W.J., "Situated cognition: On human knowledge and computer representations", Cambridge University Press, 2000.

8 Gero, J.S., "Situated design computing: Principles", Civil engineering computations: Tools and Techniques, Saxe-Colburg Publications, Stirlingshire, UK, pp. 25-35, 2007.

9 Landauer, T.K. and Dumais S.T., "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge", Psychological Review, 104(2), 211 – 240, 1997.

10 Kalman, D., "A singularly valuable decomposition: The SVD of a matrix", The College Mathematics Journal, 27(1), 2-23, 1996.

11 Pimmler, T.U., "A development methodology for product decomposition and integration", MIT Master's Thesis, Department of Mechanical Engineering, 1994.

A Problem Decomposition Methodology for Conceptual Design

This is a copy of the paper: Sarkar, S, Dong, A and Gero, JS (2009) A problem decomposition method for conceptual design, *ICoRD-09* (to appear)