Conceptual Level Design for Assembly Analysis using State Transitional Approach

Wynne Hsu	Andrew Lim	C. S. G. Lee
DISCS	ITI	Elec. Eng.
National U. of Singapore	National Computer Board	Purdue University
Singapore 119260	Singapore 0511	West Lafayette
Singapore	Singapore	IN47907, USA
whsu@iscs.nus.sg	alim@iti.gov.sg	csglee@ecn.purdue.edu

Abstract

Traditionally, design for assembly is done during the detailed design phase. A designer first maps a set of design requirements into a set of components or subassemblies that can satisfy the given set of requirements. The components and subassemblies are then examined individually to determine whether they conform to the principles of design for assembly. Usually, local changes are performed so that the resultant components/subassemblies are better for assembly. In this paper, we propose to bring the design for assembly analysis into an even earlier phase – that of the conceptual design phase. We argue that by incorporating the design for assembly analysis at the conceptual design phase, we can achieve a more substantial savings as compared to the savings obtained when the design for assembly analysis is only performed as late as the detailed design phase. The basic idea is to select a combination of design concepts (previously stored in a library) such that together they can achieve the stated functional requirements (in the form of state transitional graph) at the minimum cost for assembly. This problem of selecting the right combination of design concepts is reduced to the well-known set covering problem. With this reduction, many existing graph algorithms can be applied to aid in the design for assembly analysis.

1 Introduction

Design activity can roughly be divided into two phases: the conceptual design phase and the detailed design phase. During the conceptual design phase, the designer attempts to translate a given set of design requirements into some physical components such that the design requirements are satisfied. At this stage, the designer is not concerned with the details. Hence, the descriptions of the physical components are often vague and imprecise. Such high-level descriptions make quantitative reasoning difficult, if not impossible. As a result, computer-aided tools for conceptual design remain sparse and few.

Serious attempts have been made to support conceptual design activities. These attempts range from a fully-quantitative approach (where a design is modeled completely by a set of quantitative design constraints), to a semi-quantitative approach (where qualitative analysis is employed), to the history-based approach (where previous designs are reused if applicable). In the fully-quantitative approach, design is viewed as a constraints satisfaction problem. Friedman and Leondes [6, 7] provided a mathematical basis for a constraint theory. Light [11] looked into the relationships between constraints equations and dimensional changes. Constraint-based systems using symbolic algebraic manipulations have also been developed. Example of such systems is Gosling's MAGRITTE [8]. The semi-quantitative approach recognizes that some behavior of a continuous physical system is best described qualitatively [16]. The history-based approach emphasizes on the reuse of previous design knowledge either in the form of case representations or in the form of expert system rules. Examples of systems using this approach are the BOGART system [12], the ARGO system [9] and the CADET system [14, 15].

In spite of the increasing recognition that more support is needed for conceptual design activities, there has been little effort in integrating the later life cycle concerns into the conceptual design phase. In this paper, we examine the need to integrate assembly considerations into the conceptual design phase. We argue that it makes more sense to start the design for

[0]This work was supported in part by the National University of Singapore Research Project Grant RP940644

assembly analysis as early as the conceptual design phase because the savings that can result from such an analysis is significant. A feasible approach using the state transitional model is proposed to support the design for assembly analysis during conceptual design phase. The basic idea is to select a combination of design concepts such that together they can achieve all the user-specified functional requirements at the minimum cost for assembly. We solve the problem by reducing it to the well-known set covering problem [2, 4]. The outline of this paper is as follows. Chapter 2 discusses the need to bring assembly issues into the conceptual design phase. Chapter 3 proposes a state transitional model for modeling the behavior of a product. The framework for conceptual level design for assembly analysis is oulined. Chapter 4 describes the reduction of the problem of design concepts selection into the set covering problem. An initial study of the feasibility of using the state transitional approach is given in Chapter 5. Finally, Chapter 6 summarizes the conclusions.

2 Assembly Considerations in Conceptual Design Phase

In 1980, Boothyrod [3] first proposed the need to design a product such that it is good for assembly. He came up with a coding system whereby a designer can evaluate how good his/her design is for assembly. Following that, the need to consider assembly issues during the design phase has been acknowledged. Research in design for assembly generally takes two paths. In the one path, quantitative measures are generated to evaluate the goodness of a design for assembly. In another path, qualitative approach [1] is taken whereby high-level design principles are given to guide the design process. Automated tools to perform DFA analysis has also been built. However, all these techniques focus at the detailed design phase where analysis are carried out when the precise details become available. For example, in order to determine how good a component is for assembly, we need to know all the features in the component. A simple protrusion at the wrong place can greatly affect the assemblability index (see Figure 1).

No doubt, detailed level analysis can result in an improved design that is good for assembly. However, we believe a more substantial savings will result if we bring assembly considerations into the conceptual design phase. Some design concepts are inherently easier to be assembled than others (see Figure 2); and some design concepts may not be suitable when it is used

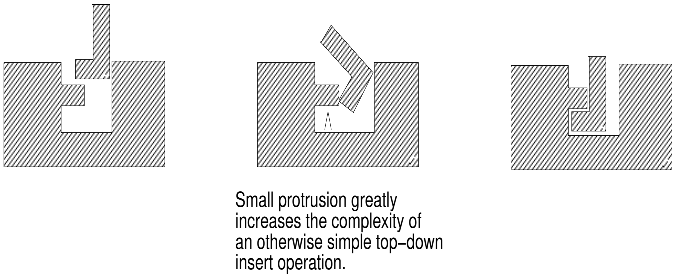

Figure 1: The Effect of a Small Protrusion on Assembly

in combination with other design concepts (see Figure 3).

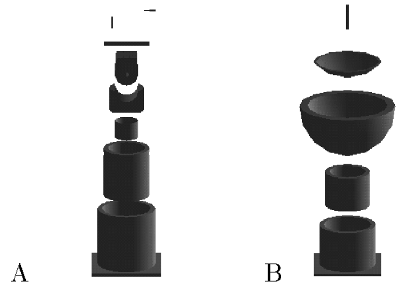

Figure 2: Two design concepts for the function "to tilt". Design concept A requires 8 components. Design concept B requires just 5 components.

By selecting the right combination of design concepts to satisfy a given set of functional requirements, substantial savings can be obtained in terms of the cost for assembly. However, though the savings are potentially great, design for assembly analysis has not yet been done systematically at the conceptual design level. One reason is because of the sheer complexity of the conceptual-level design tasks. An automated tool, that is able to systematically list all possible combinations of design concepts that can satisfy the given set of functional requirements and to give an indication of the assemblability of each design concepts, is necessary to make conceptual-level design for assembly analysis a reality.

3 State Transitional Model

To support the ability to analyze, evaluate, and select the right combination of design concepts for assembly, it is necessary to use a data representation that can model the functional behaviors of a product without dictating some preferred design concepts for

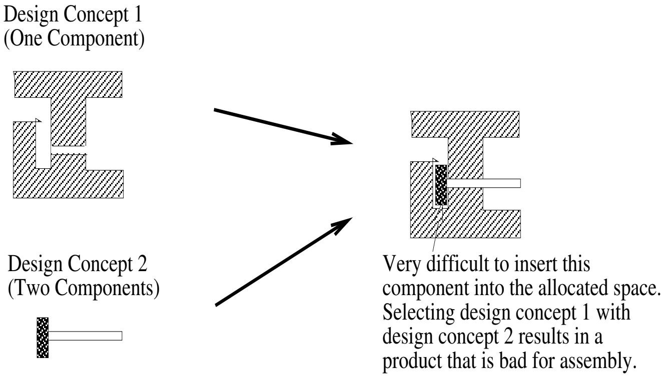

Figure 3: The Effect of Combination of Design Concepts on Assembly

realizing the functional behaviors. To this end, we propose the state transitional model representation.

A product may be viewed as a system that converts inputs to outputs. The system may go through a few states before the conversion from inputs to outputs is completed. For example, a washing machine is a system that accepts dirty clothes as inputs and outputs the clean clothes. To achieve the conversion from dirty clothes to clean clothes, the washing machines goes through a soak state, a rinse state, and a spin state. A state is generally characterized by by its inputs and its outputs. We define a state as $Z = (I|O)$ where Z is the name of the state, I is the set of possible input objects and O is the set of possible output objects. For example, the soak state for a washing machine, Z_{soak}, is defined as

$$\begin{aligned} Z_{soak} &= (\{dirty(clothes), cold(water)\}| \\ &\quad \{dirty(clothes) \wedge wet(clothes), \\ &\quad warm(water)\}). \end{aligned}$$

Graphically, we represent the states as vertices of a graph. The state transition from state s to state t is represented by a directed edge from s to t.

Mathematically, we write

$$P = (Z, F)$$

where P is the name of the product. The set Z is the set of vertices which represents all possible states of the system P. F denotes the set of directed edges representing the state transitions.

The advantages of using the state transitional model to represent the product's functional behavior are:

- the state transitional model provides a good means for modeling functional behaviors of a product without dictating a specific implementation for realizing that behavior. For example, using the influence graph model proposed by Sycara and Navinchandra[15], to specify the behavior of "to heat water", the influence graph must show how the change in one engineering variable will affect another engineering variable. One possible influence graph is

  $$R+ \longrightarrow T+$$

  where R denotes the resistance of a heating coil and T denotes the temperature of the water. Hence, in deriving the influence graph, the designer is forced to select a particular design concept to realize that behavior. On the other hand, with the state transitional model, we simply require the state transition of

  $$\text{cold water} \rightarrow \text{hot water}$$

  It does not impose any constraints on which design concepts can be used to achieve this behavior.

- the state transitional model can be represented as a directed graph. Hence, all the algorithms developed for graph theory can be applied to our reasoning process. This provides a strong theoretical basis for our work.

With this model, the framework for conceptual-level design for assembly analysis is shown in Figure 4. Our main idea is as follows. The designer will input the functional requirements of the product in the form of a state transitional graph. A X11/Motif graphical user interface is built to support this input task. This state transitional graph, denoted by G, represents the functional behavior of the product to be designed. A design concepts library, consisting of realizable design concepts for different state transitions, has been constructed. A DFA index is tagged with each design concepts stored in the library. In addition, associated with each design concept in the library is a function C denoting the integration cost. During the conceptual design phase, a systematic search is conducted. The goal is to find a subset of the design concepts stored in the design concepts library that realizes all the state transitions in G such that the cost of this subset is minimum. This problem is then reduced to the set covering problem so that we can apply the known solutions for the set covering problem to solve our design concepts selection problem. Having selected the subset of design concepts, the next task is to integrate

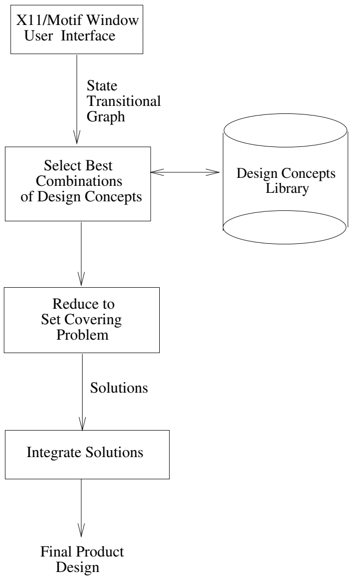

Figure 4: The Overall Framework

them into a final product. This involves resolving possible interactions among the selected design concepts. Interested reader please refer to [10].

4 Reduction to the Set Covering Problem

4.1 Design Concepts Library

From the previous section, we see that the availability of a design concepts library is vital to the proposed framework. Effort has been directed to design and construct a design concepts library. A team of 5 people, 1 computer science lecturer, 2 mechanical engineering lecturers and 2 final year mechanical engineering students, are involved in the design and construction of the design concepts library. Initially, the students are asked to write down as many useful state transitions as possible. For each state transition they have listed, we conduct a brainstorming session to derive as many different design concepts as possible that can realize the specific state transition. At the completion of the exercise, we have over 30 state transitions with over 100 different design concepts in total. During the next phase of the construction, the students formulate a geometry template for each design concept. The modeling of the geometry template is done on a ProEngineer platform. For each design concept, the assemblability index is computed based on the geometry template. At the same time, a list of preconditions are drawn up to indicate the constraints to be satisfied in order that the design concept can achieve the specific state transition. Table 1 shows a sample of the state transitions we have investigated.

States	Design Concepts	Pre-Conditions
Cold $\rightarrow$ Hot	Gas fuels	constant heat gas piping
	Solid fuels	inconsistent heat need refilling
	Solar heat	need sunlight large surface area
	Electrical coil	need electricity
	Chemical reaction	Required a confined area
Enclosed $\rightarrow$ Open	Screw cap	rigid
	Door with hinges	rigid
	Zip	non-rigid
	Tab	groove for easy removal
Wet $\rightarrow$ Dry	Hot blower	withstand high temperature
	Over bake	high temperature
	Spin dry	spinning mechanism
	Heating element	withstand temperature
Stationary $\rightarrow$ Horizontal Motion	Compressed air	rigid tank
	Compressed string	barrier
	Steam locomotive	water, fuel
	Motor	electricity
	Conveyer belt	electricity
⋮		

Table 1: Partial list of Design Concepts in the Library

4.2 The Reduction Process

Design Concepts Selection Problem:
Given a state transitional graph $G = (V, E)$ where V is the set of vertices and E is the set of edges, and a set of design concepts $D = \{d_1, d_2, \cdots, d_n\}$ with costs $w_1, w_2, \cdots, w_n$ respectively. Let g_{d_i} denote the state transitional graph of design concept d_i. Find a subset of D, D^*, such that $\cup(g_j : j \in D^*) = G$ and cost of D^* is minimum.

This problem can be decomposed into two subproblems: (1) For each g_{d_i}, determine whether it is a subgraph of G. (2) Let $D^{'}$ be the set of design concepts whose associated state transitional graphs are subgraphs of G. Find a subset of $D^{'}$ such that it covers the entire graph G with minimum cost.

Subproblem (1) is a well-studied subgraph isomorphism problem [13, 5]. The work by Eppstein [5] indicates that the subgraph isomorphism problem in planar graphs can be solved in linear time.

To solve subproblem (2), we transform the problem into the set covering problem. Before we describe the transformation process, let us first define the set covering problem.

Set-Covering Problem:
In the set-covering problem, the data consists of finite sets $P_1, P_2, \cdots, P_n$ and positive numbers $c_1, c_2, \cdots, c_n$. We denote $\cup(P_j : 1 \leq j \leq n)$ by I and write $I = \{1, 2, \cdots, m\}$, $J = \{1, 2, \cdots, n\}$. A subset J^* of J is called a cover if $\cup(P_j : j \in J^*) = I$; the cost of this cover is $\sum(c_j : j \in J^*)$. The problem is to find a cover of minimum cost. Mathematically, the set covering problem may be written as

$$minimize \sum_{j=1}^{n} c_j x_j$$

$$s.t. \qquad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad i = 1, 2, \cdots, m$$

where

$$\begin{aligned} x_j &= 1 \text{ if } P_j \text{ is part of the cover} \\ &= 0 \text{ otherwise} \\ a_{ij} &= 1 \text{ if element } i \text{ from I is an element of } P_j \\ &= 0 \text{ otherwise.} \end{aligned}$$

To transform subproblem (2) into the set-covering problem, first we construct a bipartite graph in the following manner: A bipartite graph is one whose vertex set can be partitioned into two subsets X and Y so that each edge has one end in X and one end in Y. For each state transition edge in graph G, a vertex v is created and placed in the partition X. For each d_i in set D', a vertex w is created and placed in partition Y. An edge connects the node x in partition X to a node y in partition Y if and only if the state transition associated with node x is isomorphic to an edge in the state transitional graph of the design concept associated with node y. To reduce subproblem (2) to the set-covering problem, we let the nodes in partition Y to be the finite sets $P_1, P_2, \cdots, P_n$ in the set-covering problem. $c_1, c_2, \cdots, c_n$ in set-covering problem corresponds to the assembly costs of the design concept associated with each node in partition Y. Finally, each node in partition X corresponds to an element in the set I. With this mapping, we see that a solution to the set-covering problem implies a solution to the design concepts selection problem and vice versa. Now, the set-covering problem is known to be NP-complete. A heuristics for solving the set-covering problem is used to find a near optimal solution to this problem [2]. By using this heuristic algorithm to arrive at the solution for the set-covering problem, we can solve the design concepts selection problem. The algorithm begins with an empty set as the initial solution and successively add to the solution the subset P_j which covers previously uncovered elements of I at a minimum average cost per new cover. The process continues until all the elements of I are covered by the solution set J^*. Figure 5 shows the algorithm. The time complexity of the heuristic algorithm is $O(m^3 N)$.

Step 1. Initialize: The solution set is empty.
No rows are covered.
Step 2. Select an augmenting Column:
Choose the column that has the minimum average cost per uncovered row.
Call this column j^*.
Step 3. Update the solution:
Add column j^* to the solution.
Update row coverage.
If all rows are covered, STOP.
Otherwise Go To Step 2.

Figure 5: Algorithm for Solving Set Covering Problem

5 Results of Initial Study

A prototype system is being built to demonstrate the feasibility of this approach. There are two modules to the system. The first module implements a X11/Motif-based window interface for the input of the state transition graph. Once the state transition graph has been entered into the system, a procedure that implements the graph coverage algorithm is invoked. The output of this procedure is a set of feasible design concepts that can cover the input state transition graph. The second module of the system is developed on the Pro/Engineer platform. This module implements the modeling of design concepts, the DFA analysis and the synthesis of selected design concepts.

Figure ?? gives the overall architecture of the system. Figure ?? shows the state transition graph of a chair. Figure ?? shows the retrieved design concepts based on the coverage of the state transition graph. From the figure, we see that the system is able to retrieve the essential components of a chair and that each component conforms to the principles of design for assembly. Work is now on-going to look into the issues involved in integrating these components into one functional product.

6 Conclusions

In this paper, we describe a new representation for modeling the functional requirements of a product.

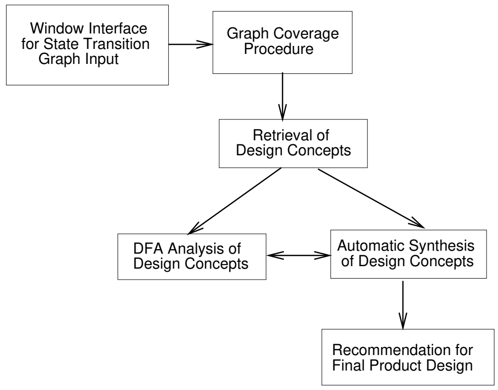

Figure 6: The Overall System Architecture

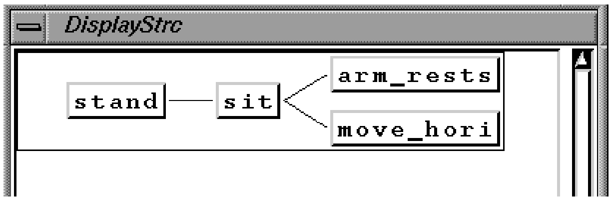

Figure 7: The State Transition Graph for a Chair

Initial study indicates that state transitional representation is capable of modeling the functional behaviors of real-life products. The framework for conceptual-level design for assembly analysis has been proposed. The design concepts selection problem is reduced to the well-known set covering problem. Additional work is needed to integrate the selected design concepts into a final product design.

7 Acknowledgement

We would like to thank Dr Jerry Fuh, Dr Y. F. Zhang, Albert Loh and Liow Kok Peng for their help in the construction of the design concepts library.

References

[1] M. M. Andreasen, *Design for Assembly*, Springer-Verlag, 1988.

[2] E. K. Baker, "Efficient heuristic algorithms for the weighted set covering problem," *Comput. and Operations Research*, 8(4):303–310, 1981.

[3] G. Boothroyd, *Design for Assembly Handbook*, University of Massachusetts, Amherst, 1980.

[4] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, 4(3):233–235, 1979.

[5] David Eppstein, Subgraph isomorphism in planar graphs and related problems, Technical Report 94-25, Dept. of Information and Computer Science, University of California, Irvine, 1994.

[6] G. J. Friedman and C. T. Leondes, "Constraint theory, part i: Fundamentals," *IEEE Transactions on Systems Science and Cybernetics*, 5:48–56, 1969.

[7] G. J. Friedman and C. T. Leondes, "Constraint theory, part iii: Inequality and diescrete relations," *IEEE Transactions on Systems Science and Cybernetics*, 5(3):191–200, 1969.

[8] J. Gosling, *Algebraic Constraints*, PhD thesis, Dept. of Computer Science, CMU, 1983.

[9] M. N. Hahns and R.D. Acista, Argo: An analogical reasoning system for solving design problems, Technical Report ML-TR-1, Department of Comp Sci, Rutgers University, 1987.

[10] Wynne Hsu, Jerry Fuh, and Y. F. Zhang, Interactive conceptual design for assembly, Technical report, National University of Singapore, 1996.

[11] Robert Light and David Gossard, "Modification of geometric models through variational geometry," *Computer-Aided Design*, 14(4):209–214, July 1982.

[12] J. Mostow and M. Barley, Automated reuse of design plans, Technical Report ML-TR-14, Department of Comp Sci, Rutgers University, 1987.

[13] J. Plehn and B. Voigt, "Finding minimally weighted subgraphs," *Proc. 16th International Workshop WG90, Graph-Theoretic Concepts in Computer Science*, pages 18–29, 1990.

[14] K. Sycara and D. Navinchandra, "Representing and indexing design cases," *Proceedings of the Second International Conference on Industrial and Engineering Applications of AI and Expert Systems*, 1989.

[15] K. Sycara and D. Navinchandra, "Index transformation techniques for facilitating creative use of multiple cases," *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 1991.

[16] Tong and Sriram, editors, *Aritificial Intelligence in Engineering Design, Volume 2*, Academic Press, 1992.