Comparative Assembly Planning During Assembly Design

Gerard Jounghyun Kim
Rm. A-127, Bldg. 220
National Institute of Standards and Technology
Gaithersburg, MD 20899, USA
e-mail: gkim@crne.nist.gov

Sukhan Lee and George A. Bekey
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0782, USA

Abstract

Recent strides toward concurrent engineering have called for a need for integrating design with assembly planning. That is, a true concurrent engineering platform must be able to, for example, perform preliminary assembly planning during conceptual design stages so that alternative assembly plans can be evaluated and compared for redesign or refining a design with a promising candidate assembly plan. This paper presents such an integrated system of an assembly planner and a (re)designer, in which, results from comparative analysis of preliminary candidate assembly plans are used to (re)design further details of a given assembly. The integration results in a more effective way of doing assembly planning and "Design-for-Assembly". A redesign process of a simple switch box is demonstrated to illustrate the benefits.

1 Introduction

Generally, the problem of assembly planning concerns determining an order (linear or partial) of assembling a product. Most usually, assembly planning is performed upon a final design, and based on the results, a need for redesign may be revealed. This is a rather inefficient and ineffective redesign process which may require repetitive and lengthy design revisions, starting from early stages of design.

The concept of assembly planning as a separate process from design is being challenged as the practice of concurrent engineering is becoming more prevalent in the American industry. A true concurrent engineering system should, if possible, allow prediction and resolution of design problems with a preliminary assembly plan before the design is finalized. Conversely, an assembly plan may change due to new constraints given by a revised design.

Closely related to the issue of achieving concurrent engineering in this context are the principles of "Design-for-Assembly (DFA)". DFA is a set of design guidelines for improving product designs for easy and low cost assembly [2]. Principles of DFA have surfaced as one of the important criteria and heuristics for determining a cost-effective assembly plan.

Figure 1 illustrates an example of (re)designing a subassembly for fewer assembly directions. Note that this particular (re)design is dependent on a choice of a particular base part and subassembly grouping (i.e. an assembly plan).

This paper presents an integrated system of an assembly planner and a DFA redesigner, called INSPIRE-2 (INIntelligent Assembly Planning Integrated with REdesign), in a continued effort to the work reported in [13]. It is more than a mere software inte-
2 Related Work:
Concurrent Assembly Design

Much work has sprung up in the field of concurrent engineering [15] [3] [4] and assembly modeling [16] [11] [18] [6]. However, very little focus has been paid to the issue of concurrent assembly design. Most concurrent engineering systems focus on individual part design and are configured as a loose coupling (e.g. by networking) of engineering softwares (e.g. FEM, product database, CAD, process planner), where it is up to the user to access and gather relevant information for his or her particular engineering purpose.

Many assembly analysis methods have been devised in connection with assembly planning systems with design feedback capabilities [6] [1] [7] [17], although it is unclear how the analysis results could be integrated with assembly design systems. Hsu et. al. developed a feedback system that used three objective criteria for evaluating an assembly sequence with regard to DFA to identify parts that need redesign, and to propose a redesign [9]. In [20], a method for incrementally analyzing an assembly during design with respect to tolerance is presented in the context of a larger concurrent engineering system called Next-Cut [4].

3 Overview of INSPIRE-2

Figure 2 shows the overall architecture of INSPIRE-2. Given input descriptions of an existing design and its assembly operations, a default design process is generated stage by stage, starting from conceptual design. The default design process is replayed from its beginning as if the product is being designed from scratch. For each design stage, a Local analysis is performed to find DFA problems that are independent of an assembly sequence. A few promising assembly plan candidates may be chosen based on a Global analysis. The Global analysis evaluates each candidate assembly sequences with respect to subassembly stability, directionality, and parallelism. A comparative analysis among candidate plans may be performed to optimize the assembly cost with respect to certain analysis parameters. Findings from the analysis are mapped to the appropriate portion of the design process. The design process is replayed and whenever a problematic design decision is encountered, it is modified using a case-based approach.

When all the design problems in a given design stage are addressed, the design process for the next design stage is generated and the same process continues. When the system reaches the final design stage,
an output in the form of a new design process and a new assembly plan is produced. For a better understanding of the details of BAP and REV-ENCE, please refer to [12] and [10].

4 Interleaving Assembly Planning and (Re)Design

4.1 Design Representation

A Design Object refers to any object whose existence or form is to be determined by a designer and includes Artifact and Operation. An Artifact is a collection of design objects that represent the assembly. Note that an assembly operation is regarded as a design object as well. Each design object is associated with design decisions, which create and set various attribute values of the design object, and define its relations to others. Each design decision may be supported by number of design rationales, and marked with design problems (after analysis). Design problems may map to redesign cases that might be able to solve them. An example assembly of a switch box is shown in Figure 3.

4.2 Generating Default Design Process

Once descriptions of a design and associated assembly operations are entered into the system, the next step is to generate a default design process in order to start the process of (re)design by replay and modify [14]. A default design process is defined as a probable sequence of design decisions and their justifications that could have created the given design.

Vocabularies describing different design decisions are predefined in terms of a precondition-and-effect operators. Different types of design decisions are grouped into different decision classes, starting from functional design to detailed form design, and further subdivided into 11 design stages as shown in Table 1. Each design stage is composed of series of design decisions of which are responsible for filling in certain details of the design appropriate for its stage. A heuristic algorithm, based on a generic design process model, selects and schedules probable design decisions, and constructs design states associated with them. The reconstruction starts from a default null initial design state, and ends at a final state where all of design objects and design information have been accounted for. The detailed design process model is shown in Table 1.

4.3 Incremental Design Analysis

A redesign first proceeds by incrementally generating each stage of the design process and perform-
4.3.2 Global DFA Analysis

On the other hand, a global analysis refers to a process of identifying any DFA problems that are assembly sequence dependent. Therefore, a global analysis involves performing assembly planning and selecting preferred subassemblies. When subassemblies are selected, design decisions involving them are generated and analyzed, as if they were individual parts/features in the manner described in the preceding subsection. Problematic design decisions for subassemblies may include those responsible for missing guidance features, unstable subassemblies, and high number of assembly directions.

The following conditions (subset of Lee's conditions [12]) are used to identify feasible subassemblies for producing all possible subassemblies. For a cluster of parts, S, that belongs to an assembly A (denoted S[A]) to be a subassembly of A if, (1) S[A] can be brought to A - S[A] from free space for mating, (2) interconnection defined between S[A] and A - S[A] is feasible, (3) all mating between S[A] and A - S[A] have at least one common axis of separation, and (4) S[A] and A - S[A] are stable either by itself or by external devices. Conditions 1 and 2 are computed automatically using a non-guaranteed algorithm based on simple interference check along a projected path in six principal directions. Condition 3 is checked by querying a knowledge base of various interconnections, while condition 4 is manually checked.

Then, an AO* search is performed to find the preferred subassemblies. The global DFA analysis, therefore, corresponds to the heuristic function used in the AO* search described in [12]. In this paper, we introduce a simpler version of Lee's heuristic function [12] that does not include local cost (c₁ and c₂), since local analysis is provided separately as described in Section 4.3.1. The search space to which the AO* algorithm is applied can be represented by an AND/OR tree. The decomposition of an assembly A implies the expansion of an AND node (representing an assembly A) into its OR children representing the alternative decompositions of A. Figure 4 shows a AND/OR tree for feasible subassemblies of the switch box example. A potential solution tree is an AND tree having the minimum value for the evaluation function at the current stage of search, whereas a solution tree is an AND tree with leaves consisting of only single parts.

To formulate the evaluation function, eₘ for the AO* algorithm, let us introduce the following definitions:

Definition: The Heuristic Estimate, hₑ(n), associated with an OR node, n, represents an estimate of the optimal relative assembly cost to assemble S[A] and A - S[A], and can be computed by the weighted sum of the following components.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Design Decision</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>select node</td>
<td>Create subassembly related decisions.</td>
</tr>
<tr>
<td>2</td>
<td>create form</td>
<td>Create assembly related decisions.</td>
</tr>
<tr>
<td>3</td>
<td>assemble assembly</td>
<td>Create assembly related decisions.</td>
</tr>
<tr>
<td>4</td>
<td>create assembly</td>
<td>Create assembly related decisions.</td>
</tr>
<tr>
<td>5</td>
<td>assign function</td>
<td>Assign function to assembly related decisions.</td>
</tr>
<tr>
<td>6</td>
<td>assign function</td>
<td>Assign function to assembly related decisions.</td>
</tr>
<tr>
<td>7</td>
<td>assign function</td>
<td>Assign function to assembly related decisions.</td>
</tr>
<tr>
<td>8</td>
<td>assign function</td>
<td>Assign function to assembly related decisions.</td>
</tr>
<tr>
<td>9</td>
<td>assign function</td>
<td>Assign function to assembly related decisions.</td>
</tr>
<tr>
<td>10</td>
<td>assign function</td>
<td>Assign function to assembly related decisions.</td>
</tr>
<tr>
<td>11</td>
<td>assign function</td>
<td>Assign function to assembly related decisions.</td>
</tr>
</tbody>
</table>

Table 1: Design process in INSPIRE-2.

Then, an AO* search is performed to find the preferred subassemblies. The global DFA analysis, therefore, corresponds to the heuristic function used in the AO* search described in [12]. In this paper, we introduce a simpler version of Lee's heuristic function [12] that does not include local cost (c₁ and c₂), since local analysis is provided separately as described in Section 4.3.1. The search space to which the AO* algorithm is applied can be represented by an AND/OR tree. The decomposition of an assembly A implies the expansion of an AND node (representing an assembly A) into its OR children representing the alternative decompositions of A. Figure 4 shows a AND/OR tree for feasible subassemblies of the switch box example. A potential solution tree is an AND tree having the minimum value for the evaluation function at the current stage of search, whereas a solution tree is an AND tree with leaves consisting of only single parts.

To formulate the evaluation function, eₘ for the AO* algorithm, let us introduce the following definitions:

Definition: The Heuristic Estimate, hₑ(n), associated with an OR node, n, represents an estimate of the optimal relative assembly cost to assemble S[A] and A - S[A], and can be computed by the weighted sum of the following components.

Definition: The Heuristic Estimate, hₑ(n), associated with an AND node, n, represents an estimate of the optimal relative assembly cost to assemble S[A] and A - S[A], and can be computed by the weighted sum of the following components.

# AND tree in an AND/OR tree where every AND node has no more than one OR child

322
The relative assembly cost, \( R_2(S[A]) \) and \( R_4(A - S[A]) \), due to the number of orientations involved in the assembly of \( S[A] \) and \( A - S[A] \).

The relative assembly cost, \( R_3(S[A]) \) and \( R_4(A - S[A]) \), due to the number of stacking and non-orientation operations involved in the assembly of \( S[A] \) and \( A - S[A] \).

The relative assembly cost, \( R_5(S[A]) \) and \( R_4(A - S[A]) \), associated with the stability of \( S[A] \) and \( A - S[A] \).

\[
I(S[A]) = \frac{\text{no. of floating liaisons in } S[A]}{\text{ave. deg. of a node in } S[A]}.
\]

The effect of parallelism, \( w_p(S[A]) \), where:

\[
w_p(S[A]) = \frac{N_1 - N_2}{N_1 + N_2},
\]

where \( N_1 \) and \( N_2 \) represent the number of parts in \( S[A] \) and \( A - S[A] \), respectively.

**Definition:** The evaluation function, \( e_f(T) \), associated with an AND tree \( T \), is equal to sum of \( h_e(n) \) for all \( n, n \in T \).

The content of the input representation for both design and assembly operation is assumed to be obtainable except for attribute values that are not usually available from standard design documents, including DFA characteristics of parts and subassemblies such as handling and orienting costs, stacking, and part symmetry. The values of these attributes are obtained from the user during the analysis process using a query/answer form (similar to those of Boothroyd DFA software and Sapphire's Assembly View [2] [18]) in order to compute the heuristic function.

### 4.3.3 Comparative Analysis

Instead of just selecting the current best assembly plan and continue to (re)design for it, several candidates for assembly plans may be kept and analyzed with respect to an evolving design. For instance, the second best assembly plan must be better than the best in terms of number of fixtures required, but only worse in terms of number of stacking operations. Each assembly plan candidate requires different design strategies and, therefore, after redesign, a previously “worse” assembly plan may become “better”. Therefore, keeping a multiple number of assembly plans and (re)designing them involves keeping multiple threads of (re)design processes.

Figure 4 shows an AND/OR graph generated right after design stage 9. It contains two feasible subassemblies for the switch box example (other possibilities are omitted). The first possibility (upper half of the AND/OR graph), hence called plan 1, is to assemble one part at a time, and the other possibility (lower half of the AND/OR graph), hence called plan 2, splits the assembly into two groups of subassemblies. At design stage 9, plan 2 is better than plan 1 as indicated by the lower \( e_f \) value shown on the top OR node of the corresponding AND/OR subtree.

For a comparative analysis, one analysis with respect only to directionality (stacking) and another with respect to subassembly stability are performed. Figure 5(a) and Figure 5(b) shows the resulting AND/OR graphs respectively. Figure 5(a) shows that plan 1 is as good as plan 2 as far as directionality is concerned, although it contains two unstable subassemblies, namely due to two floating liaisons between the case and two contacts, contact1 and contact2. This is automatically found by examining the corresponding AND/OR subtree and looking for a significant jump (arbitrarily defined) in the accumulated heuristic measure, \( e_f \). Appropriate problem descriptions are generated and attached to the corresponding responsible design decision. Figure 6 shows problems mapped on one of the corresponding design decisions select-form-case-contact1-liaison. Note that these problems are only relevant and meaningful in the context of using plan 1.

### 4.3.4 Redesign

As for the switch box example, there will be two separate processes for redesign, since two distinct assembly plans are being considered in this particular example (plan 1 and plan 2).

After the global comparative analysis, it is found, for plan 1, that two loose mating relationships (floating liaisons) between the case and metal contacts are causing instability in the respective subassemblies.

A simple redesign is proposed and executed interactively by the case-based redesigner to change the type of liaison to a force-fit (instead of insertion) to provide stability. A new assembly plan evaluation is performed. As shown in Figure 7, plan 1 is now as good as plan 2 with regards to a global analysis.

Other problems that are found and solved for each redesign process are summarized in Table 2 and Figure

An assembly is represented by a liaison graph, where nodes correspond to parts, and arcs correspond to liaisons or mating relationships among parts. A non-floating liaison requires external connectors, forces or mechanisms for interconnecting the associated parts, while a floating liaison does not.
Table 2: Problems and proposed solutions for plan 1 and plan 2.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Plan 1</th>
<th>Plan 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interfering mating lines</td>
<td>Snap fit</td>
<td>Snap fit</td>
</tr>
<tr>
<td>Interfering mating lines</td>
<td>Large feature size</td>
<td>Snap fit</td>
</tr>
<tr>
<td>No guidelines indicate the assembly sequence</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that the second and third redesign solutions in Table IV are only applicable in the context of sub-assembly groupings of plan 1. Widening of slot sizes in the panel for plan 2 would make the subassembly of panel and metal contact unstable.

5 Conclusion

In this paper, we have presented an integrated system of (re)design and assembly planning, called INSPIRE-2. INSPIRE-2 has been implemented in COMMON LISP (about 15,000 lines of code). INSPIRE-2 is more than a mere software integration, since the activities of assembly planning and (re)design are interwoven during the (re)design process under a single framework.

Interweaving preliminary assembly planning with design can result in a reduction of cycle time between artifact design and process design. According to Kashevan [9], manual assembly planning usually goes through many iterations (typical 6-10), and plan maintenance (in response to design changes) is about 5 times the cost of plan generation. Traditionally, process design follows after an artifact design is completed. In a worst case scenario, this results in a vicious cycle of requests for design revisions and degradation in the level of automation.

Although conventional DFA methods have proven to be effective analysis tools, they do not provide a framework for considering alternative assembly plans and subassembly groupings. Consequently, a correct analysis may not be possible since some DFA criteria must be measured with respect to an actual assembly sequence. Combining design and assembly planning is a natural step to both realize a more global DFA analysis and effective redesign strategies.

Certainly, there are still many number of problems that need to be resolved in order to further enhance the system in terms of its usability and effectiveness.

Current research efforts are focused on improving and expanding representations, particularly for describing mating relationships and assembly design operators, automating the process of determining motion constraints, subassembly poses and stability to reduce number of required user queries and providing visual and geometric feedback for newly proposed designs.

References


Figure 4: An AND/OR tree for feasible subassemblies of the switch box example. Only two possible subassembly grouping plan 1 and plan 2) is shown. For each OR node, the corresponding subassembly pairs and heuristic measures are printed.

Figure 5: (a) An AND/OR tree for feasible subassemblies of the switch box example. Analysis is performed only with respect to directionality, that is, how stacking the subassemblies are. (b) An AND/OR tree for feasible subassemblies of the switch box example. Analysis is performed only with respect to instability of subassemblies.
Figure 6: Problematic design decisions in design stage 7 marked by the comparative analysis.

Figure 7: An AND/OR tree for feasible subassemblies of the switch box example after redesign. At this point, plan 1 is as good as plan 2.

Figure 8: Two possible redesigns for two assembly sequence candidates realized with Pro/Engineer CAD systems following redesign advice from INSPIRE-2. In the end, plan 1 results in lower overall cost because it no longer requires fixtures.