UPGRADING DESIGN SYSTEMS

Sarosh Talukdar, Jim Rehg, Rob Woodbury, Alberto Elfes

Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Design systems can have considerable embedded value. Improvements in such systems are better achieved through upgrades than through complete replacements. To determine how best to make these upgrades requires a systems view of design. Such a view is provided by what we call TAO (test-aspect-operator) graphs. Nodes in these graphs represent aspects of the artifacts being designed while arcs represent operators (transforms between aspects) and tests (comparisons of aspects). Upgrades can be thought of as the additions of nodes or arcs to an existing TAO graph. To illustrate these ideas we will briefly describe the upgrades that we are making to a system for designing certain automobile parts.

1 Introduction

The important properties of artifacts like cars, computers and microelectronic chips are determined by their designs. Manufacturers of such artifacts must maintain competitive design systems, if they are to preserve their market shares. When design systems have considerable embedded value, as is often the case, it makes more sense to maintain their competitive margins through frequent and relatively small upgrades, rather than through infrequent and massive changes.

To understand where upgrades are most needed we must identify the principal weaknesses of existing design systems. Consider the three stages of virtually every large design project, namely: the decomposition of the project into loosely coupled, partially ordered tasks, the performance of the tasks, and the integration of the results. For the third stage to work, the second stage must produce results that are compatible. One way to do this is to establish a team of agents whose function is to monitor the tasks of the second stage and make sure their results are compatible. We will refer the activities of this team as "simultaneous engineering" or "concurrent design". We note that simultaneous engineering is closely related to distributed problem solving and to ideas of contingency theory as used in human organizations. These relationships are further discussed in [5].

1.1 Second Stage Weaknesses

Some of the more profound weaknesses of existing design systems stem from their second stages. These weaknesses include:

- closed design processes that inhibit the monitoring and mid-course adjustments needed for effective simultaneous engineering;
- limited vision and delayed feedback (The decisions made in upstream design tasks can have significant effects on downstream tasks. However, decision makers often cannot see these effects. When they can, it is through simultaneous engineering and tests that are usually so slow that it is difficult to do anything about the effects.);
- weak infrastructures that can support only a fraction of the representations needed (Complex design processes call for wide varieties of representations. Various geometric models, differential equations and block diagrams are some examples. Existing CAD systems, however, support only a very few of these representations.);

2 Design Processes

2.1 Process Architectures and TAO Graphs

The three main components of a design process are aspects, operators and tests. An aspect is a perspective or view of an artifact, from any point in its life cycle. For example, sketches on the back of an envelope, detailed blueprints, and full size prototypes are all aspects of a car. Such aspects can be divided into three categories:

- input aspects (given data),
- output aspects (goal states), and
- intermediate aspects (subgoal states or stepping stones to the outputs).

Operators calculate the intermediate and output aspects and can be either manual or automatic. Large processes usually require some of each. Tests are special cases of operators. The purpose of a test is to compare two or more aspects for consistency and post the results of the comparison in another aspect.

A convenient way to describe the architecture of a design process (the arrangement of aspects, operators and tests) is by a directed graph whose nodes represent aspects and whose arcs represent operators and tests. We will call these graphs TAO (test-aspect-operator) graphs.

2.2 Control

The control problem can be stated as follows: given a TAO graph, select paths by which to calculate the output aspects from the input aspects. Some of the given inputs may be test results. That is, the paths may have to be selected so that certain tests are passed. Often this requires cycles through the graph (iterations).

The functions of the control scheme are to produce an "initial design" by instantiating the empty aspects, and then, to reduce inconsistencies among the aspects, much as a servo control system acts to reduce errors.

The human members of a design task force usually work in a distributed mode that allows for both spontaneous and preplanned collaboration. The computer tools in existing systems, however, are usually incapable of spontaneous or opportunistic action.

2.3 Eliminating Weaknesses

What are the causes of the second-stage-weaknesses listed earlier to be alleviated? We suspect that the causes in many, if not most, industrial processes are architectural in nature. Factors to be considered in eliminating these causes are:

1. aspect placement (Do the aspects decompose the overall task into manageable subtasks? Do they provide the information and connection points needed for simultaneous engineering?);
2. aspect capability (Do the data abstractions and representation schemes used by each aspect adequately capture and present the information needed by the operators that deal with the aspect, and just as important, hide the information that might confuse these operators?);
3. test adequacy (Do the tests make sense? And are they appropriately placed? Do they cover all the consequences the designers should know about? Can they be run fast enough to provide useful feedback to the designers?);
4. modularity and expandability (Can new aspects, operators and tests be readily added to the architecture?).

3 CASE: A Project to Upgrade Window Regulator Design

It is difficult to work on issues of upgrading processes in the abstract. To provide an instance of a real, working process, CMU and Fisher Guide have combined forces and selected the design of window regulators as a prototypical process. A window regulator is a device that raises and lowers the glass in an automobile door. The type of regulator considered here has three main parts: a lift arm to move the glass; a combination of a handle, sector and pinion to translate handle rotations to lift arm movement; and a backplate to attach the entire device to the inner door panel.

A TAO graph for the existing window regulator design process is shown in Fig. 1. An examination of the graph reveals several typical flaws, including:

- Most of the operations are manual.
- Much of the generative reasoning is done in a single step (stick model to blueprints). This reasoning process is relatively inaccessible to outsiders and even the designer can forget why he did things.
- There is little feedback of the consequences of design decisions to the designer. Where there is feedback (e.g. from the lab tests and finite element analysis), it is delayed by days or even weeks.

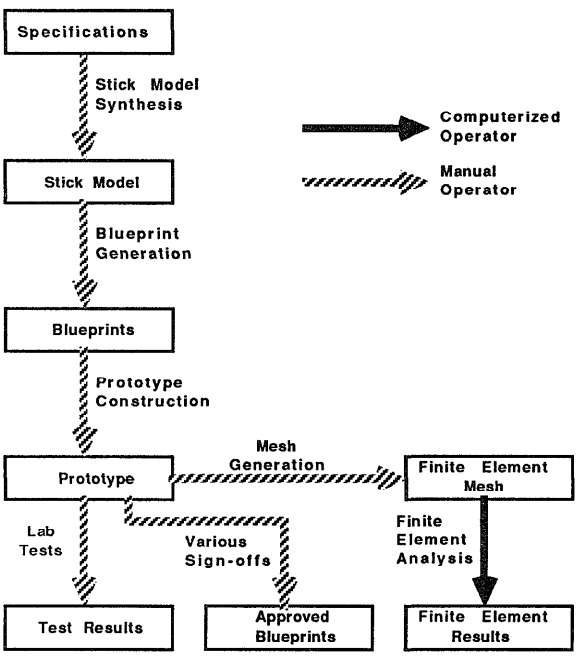

Figure 1: Test-Aspect-Operator (TAO) Graph for the existing window regulator design process.

Our plan for upgrading the process involves two phases. The goals of the first phase are to automate much of the routine, well understood parts of the existing process and to add some feedback mechanisms (tools that can quickly predict the consequences of design decisions) in important domains such as manufacturing cost. Thus, when the first phase is completed, the human designers will have a system that can

quickly transform specifications into a finished design and also provide evaluations of the design in a number of critical domains.

The goals of the second phase are to integrate the human designers into the system by providing them with a powerful interface and the means to guide the automatic operators.

Work on the first phase is nearing completion. A TAO graph for the system is shown in Fig. 2. The system involves upgrades in each of the four important areas of engineering design: synthesis, analysis, optimization and simultaneous engineering.

In the rest of this paper we report on these components, dwelling most on synthesis because it provides the backbone of the system.

4 The Design Synthesis Task

Synthesis in CASE is based on a window regulator design scenario in which existing backplate and sector designs are chosen from a parts library to meet a given set of requirements, while the lift arm is designed "from scratch", along with some smaller components, to interface to the existing backplate and sector and meet the specifications. This scenario is analogous to actual design practice in many segments of the automotive industry.

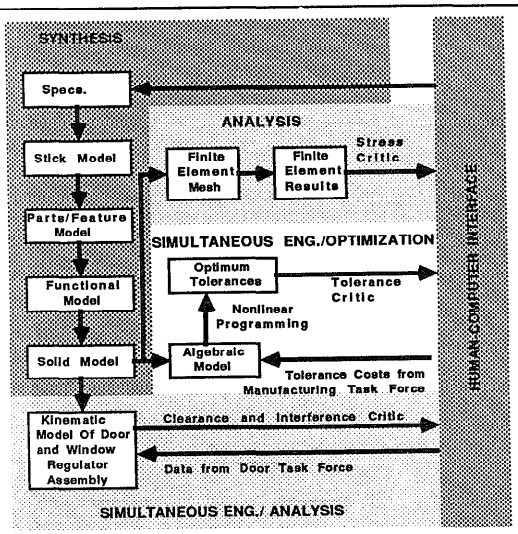

Figure 2: Test-Aspect-Operator (TAO) Graph for the improved window regulator design process.

In addition, the lift arm itself is typically one of several predetermined types, making lift arm synthesis a problem of selecting the appropriate structure for the arm and then choosing the dimensional parameters to meet the specifications. Thus, the window regulator design task falls into the class of problems often referred to as *routine design*, for which some solution methods have been proposed [1].

4.1 Design Aspects

In the paradigm described above, design synthesis consists of two activities: the selection of a group of primitives that meet the design performance requirements, and the instantiation of a set of parameter values that meet the design constraints. In routine design, the correct primitive group is assumed to be known in advance and the synthesis activity essentially involves constraint satisfaction in multiple representations. Two of the representations employed in synthesis are described below.

4.1.1 Stick Model

The stick aspect corresponds roughly to the *planar kinematic diagram* commonly employed in mechanism design [2]. It captures the basic skeleton of the mechanism and those key parameters that determine its motion. Within this representation, the synthesis task consists of choosing the major link dimensions and gear ratios necessary to meet the design specifications. As a result of the choice of stick aspect primitives, essential device parameters can be determined without considering the full detail of a manufacturable part.

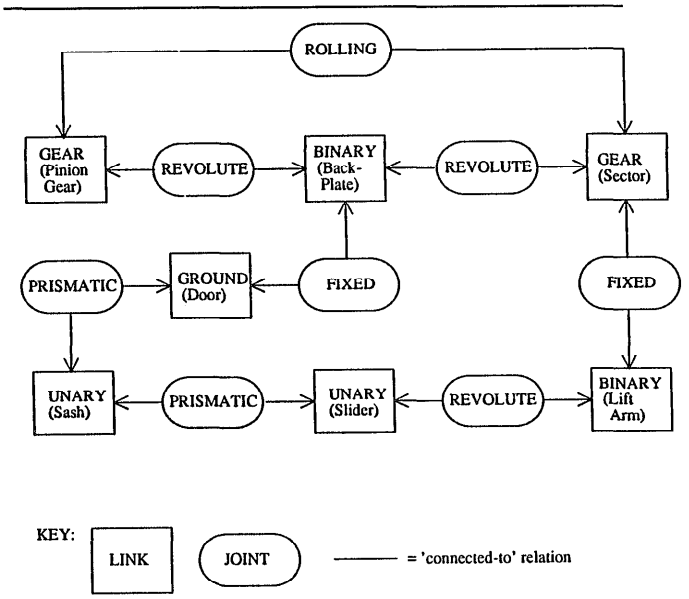

Figure 3: Stick Aspect Network. The above data structure describes the manner in which the stick aspect primitives are interconnected. The resulting graph can be traversed to generate design equations or query about the kinematic properties of the mechanism.

Following the representational paradigm described earlier, the stick model is composed of two groups of primitives: *links* and *joints*. *Links* define the skeletal structure of the mechanism, while the *joints* define permissible relative motions of links. There are three types of link primitives: binary, gear, and ground links, and four types of joint primitives: revolute, prismatic, rolling, and fixed joints. The stick model consists of a network of interconnected link and joint primitives. A sample stick aspect network is given in Fig. 3. The stick diagram described by the network is depicted in Fig. 4.

4.1.2 Parts Aspect

While the stick aspect captures the essential kinematic information about a design object, the *parts aspect* provides a description of the object *at the level of detail necessary to manufacture it*. Unlike the stick aspect, the choice of manufacturing process is important at the parts level, for it determines the types of primitives that will be employed. For example, because the window regulators are manufactured through a progressive die operation, the parts primitives consist mainly of formed sheet metal objects and rivet-type connectors.

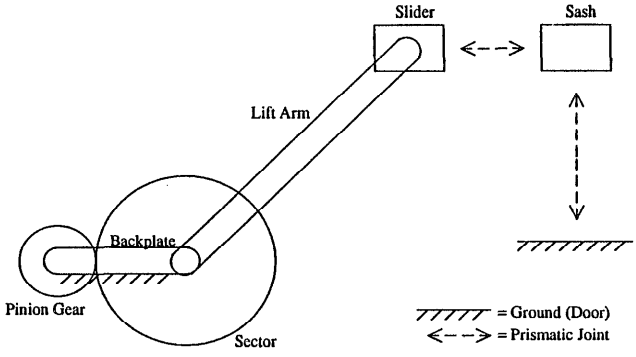

Figure 4: Stick Diagram. The above sketch depicts the stick model window regulator described by the graph of Fig. 3. Note how unnecessary design detail has been suppressed.

Although the parts aspect is more domain specific than the stick aspect, it is identical in form. The two classes of primitives it employs are *parts*, which consist of the manufacturable elements necessary to design the mechanism, and *connections*, which represent the specific fastening technologies employed in assembling the device. The part and connection primitives necessary to describe a manual window regulator are discussed in [3]. Like the stick aspect, the parts aspect has a network structure.

Unlike the stick aspect, primitives in the parts aspect are not parameterized at the parts level. Instead, each part has a *feature representation*, which describes the part as a combination of more detailed primitive design elements. In the current implementation, the development of feature representations has been restricted to the lift arm. Other parts, such as the backplate and sector, are characterized at the feature level by a single parameter list.

The feature representation of the lift arm is based on three classes of primitives: *slabs*, *formations*, and *seams*. The body of the arm is composed of slab primitives, joined together by either *flat* or *bend* seams. Flat seams are used when two slabs that lie in the same plane joined together, while bend seams are used where a change in vertical orientation occurs. Viewed from the side, the lift arm consists of alternating parallel and slanted sections connected by flat and bend seams, respectively. Each individual slab can in turn contain any of the two formation primitives: *holes* and *slots*.

In a manner similar to the stick primitives, the feature primitives are arranged in a semantic network. There are two possible network connections: *connected-to* and *contains*. As before, the primitives are characterized by a parameter set. Fig. 5 shows the top view of the arrangement of primitives that form the lift arm.

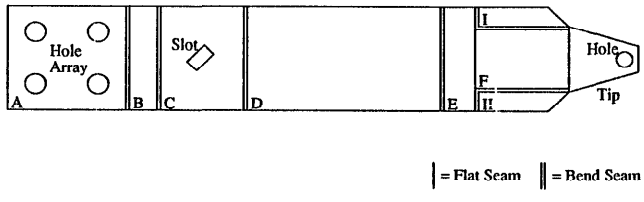

Figure 5: Lift Arm Feature Diagram. The above illustration shows the manner in which the feature primitives are arranged to model a lift arm.

4.2 Operation of the Synthesis Architecture

In the current implementation of CASE, synthesis occurs in two stages. In the first stage, the *stick model synthesis program* solves the stick aspect constraints to obtain a set of stick aspect parameters that meet the design specifications (see Fig. 2.) The stick synthesis module produces a skeletal window regulator design, in which the major design decisions have been made. The design, however, lacks the detail that would make it a manufacturable part.

In the second stage, the remaining design detail is added by the *feature model synthesis program*, which solves the feature model constraints to obtain the feature model parameters. After the feature synthesis program has performed its task, the design is complete and ready for analysis.

Note that within each design stage the problem-solving process is the same, but the design representations differ in the view of the design object they present. This separation permits the design process to occur in a hierarchical fashion that strongly resembles actual design practice. While the current design process is *sequential*, we are presently developing a more flexible problem-solving approach in which the stick and feature synthesis modules can interact in producing a design.

Located between the stick and feature aspects in Fig. 2 is a *design translator* module that *maps the effects of design decisions made in the stick representation to constraints on design decisions made in the feature representation.* The presence of the translator is necessary to ensure that, for example, the length of the feature model lift arm is compatible with the length of the binary link that corresponds to the lift arm in the stick representation. A description of the operation of the translator module is given in [4].

4.3 The Synthesis Operator

The problem-solving technique employed by the individual synthesis modules is an adaptation of the *agent hierarchy* approach suggested in [1]. In CASE, groups of design parameters are "assigned" to problem-solving agents that contain the domain knowledge necessary to generate parameter values through the use of heuristic rules and constraint propagation. These agents are arranged in a hierarchy, and communicate through message-passing. There are two basic mechanisms for assigning values to parameters: instantiation from default values and constraint satisfaction. The constraint satisfaction algorithm employs breadth-first search and is guided by domain knowledge stored in the agents as plans. This synthesis approach facilitates the incremental development of the system and provides a well-organized structure for the acquisition of domain knowledge. A more detailed description of the operation of the synthesis modules is given in [4].

5 Simultaneous Engineering

Simultaneous engineering addresses two fundamental issues in the design of any complex system. On one hand, the different, often conflicting requirements imposed on the design of a device from the point of view of the various design stages have to be addressed and harmonized. These include satisfaction of the original specifications, materials issues, handling manufacturing and assembly concerns, etc. On the other hand, a complex device will have many subsystems that are designed separately and have to be integrated into a working whole. In this Section we briefly discuss two parts of our system which support simultaneous engineering: tolerance optimization and interference/clearance analysis.

5.1 Optimization of Tolerances

The goal of tolerancing aspects and operators is to help the rational selection of tolerances based on considerations of cost, performance specification and sensitivity. When tolerance analyses are performed early in the design process, feedback from these analyses can be constructively used to guide design decisions. Tolerances may be optimized for cost, subject to performance, but this requires information that is often not available until later in design. Part of the necessary information for tolerance optimization, the device geometry, is available quite early, and may be used to examine, in a qualitative sense, the sensitivity of performance to design decisions. We report here the optimization problem and its component sub-problem, the computation of sensitivity information from design geometry.

To obtain the optimal tolerance settings, three types of input information are required:

1. The cost of holding tolerances, or controlling variations in dimensions, during manufacturing.
2. The cost associated with the degradation of a design's performance measures.
3. The sensitivity of performance measures to variations in nominal part dimensions.

Once this input information is obtained, an optimization problem is formulated to compute estimates for the optimal tolerances. Individual dimensional variances are selected to minimize the total cost of controlling these variances and the expected value of the quality loss function. The expected loss is transformed from a function of performances to a function of variances using the sensitivity relationships, the quadratic nature of the loss function, and the assumptions of normal distributions for the dimensional deviations.

The cost of holding tolerances and the cost of performance measure degradation are relatively independent of decisions made during the design process. These costs are derived from data about the manufacturing facility. They include information about rework, scrap, repair and warranties. On the other hand, the sensitivity information is extracted from design representations generated during design synthesis. For these purposes, a device is represented abstractly, as a linked series of homogeneous coordinate transformations and points. This abstract representation is automatically computed from a design representation by traversing a connection graph between parts. The rotation and translation parameters of the transforms and points describe the device dimensions. Each parameter is represented as a nominal value and an error distribution about the value. Distance functions between

points are used to express performance measures. The sensitivities of performance measures to errors are computed at the nominal parameter values by a combination of symbolic differentiation and matrix concatenation.

5.2 Interference and Clearance Analysis

The purpose of interference and clearance analysis is to verify that the separately designed components of a design do not interfere spatially, either as they are placed in the assembly or move through their paths in space. The facility consists of one test (for the dynamic interference analysis), two aspects (functional and 3D solid, which comprise a model for the analysis) and two operators (which build the aspects from the feature aspect).

5.2.1 Functional and Solid Aspects

The functional aspect represents the kinematics of the window regulator design at the feature level. It is differentiated from the stick aspect described earlier by its inclusion of specific part geometry which makes the spatial analysis meaningful.

The three-dimensional solid aspect is used to test a particular configuration of solid representations for interference and proximity. An instantiation of this aspect consists of a set of (location, solid) pairs. The solids are generated by procedural calls to a solid modeller (VEGA), made at the time of creation of the functional aspect. The locations are specified dynamically from the state of the functional aspect. The 3D solid aspect serves two purposes in the system: precise and full representation of spatial position and geometric relationships, and visualization of designs to the user interface.

6 Conclusions

The work that we have done so far falls into two categories: identifying crucial issues in upgrading design systems, and assembling apparatus with which we can begin to study these issues.

The system that has resulted is noteworthy for its openness, for its use of multiple representations and for its ability to integrate diverse types of design tools. At present, the system's operational capabilities are as follows. Given a set of specifications, the automatic parts of the system can synthesize a sketch (stick diagram), a parts/feature model (a more detailed representation than a sketch) and a solid model (a full description of the geometry) of a window regulator that meets the specifications. Through a combination of human and programmed activity, the system can also optimize dimensional tolerances (and thus, solve a large part of the design for manufacturability problem for the regulator); point out incompatibilities that may be developing between the regulator and door geometries (the door is designed concurrently with the regulator); and draw attention to mechanical weaknesses in the regulator.

Our plans for the immediate future include:

1. adding more automatic operators to the system so that most aspects in Fig. 2 can be reached by several different paths and implementing these operators in a distributed network of computers;
2. studying ways for editing designs and rapidly attenuating the inconsistencies that tend to emanate from the site of a change;
3. studying ways to rapidly reconfigure design systems.

References

[1] Brown, D. and B. Chandrasekaran.
Knowledge and Control for a Mechanical Design Expert System.
Computer :92-100, July, 1986.

[2] Erdman, A. and G. Sandor.
Mechanism Design: Analysis and Synthesis.
Prentice-Hall, Englewood Cliffs, 1984.

[3] Rehg, J., Elfes, A., Talukdar, S., Woodbury, R., Eisenberger, M. and Edahl, R.
CASE: Computer-Aided Simultaneous Engineering.
In *Proceedings of the 1988 AI in Engineering Conference.* Computational Mechanics, Stanford, August, 1988.

[4] Rehg, James M.
Computer-Aided Synthesis of Routine Designs.
Master's thesis, Carnegie Mellon University, June, 1988.

[5] Talukdar, S., Elfes, A. and Papanikolopoulus, N.
Concurrent Design, Simultaneous Engineering and Distributed Problem Solving.
In *Proceedings of the 1988 AAAI/AI in Design Workshop.* AAAI, AAAI-88, Minneapolis, MN, August, 1988.
Submitted.