Proceedings of
The 1996 ASME Design Engineering Technical Conferences and
Computers in Engineering Conference
August 18-22, 1996, Irvine, California

96-DETC/DTM-1228

CONSTRAINT MANAGEMENT METHODOLOGY FOR CONCEPTUAL DESIGN TRADEOFF STUDIES

Sudhakar Y. Reddy, Kenneth W. Fertig and David E. Smith
Science Center, Palo Alto Laboratory
Rockwell International Corporation
444 High Street, Suite 400, Palo Alto, California 94301, U.S.A.
Phone: (415) 325-1892 Fax: (415) 325-2007 E-mail: sudha@rpal.rockwell.com

ABSTRACT

This paper presents a constraint management methodology, which facilitates tradeoff studies during conceptual design. This approach represents design models as constraints between variables, and uses the resulting constraint network to automatically derive computational procedures for performing user-specified tradeoff studies. By decomposing large constraint networks into smaller pieces that can be solved robustly, this approach can solve extremely large systems of non-linear equations present in practical system models. Design Sheet is a software implementation of this methodology; it allows the designer to interactively develop models, flexibly define tradeoff studies, and quickly explore large areas of design space to study how the different performance and cost criteria tradeoff with respect to one another. Design Sheet has been used on practical applications ranging from the system-level design of spacecraft using combined performance and cost models to the preliminary design of automotive bearings. This paper demonstrates the unique capabilities of Design Sheet in performing design tradeoff studies, using a thermal imaging system performance model developed for the DARPA MADE program.

INTRODUCTION

The design of complex systems requires assessing a wide range of alternative designs, so as to determine one that best meets competing requirements from multiple perspectives, such as performance, manufacturability, reliability and cost. As detailed evaluation of large numbers of design alternatives is not practical because of resource constraints, a hierarchical approach is used in deciding the design specifications. During the early conceptual phases of this process, decisions are made about which configurations are better than which others. During the detailed phases, when the major configuration options and design choices have already been made, the task is to determine the optimal values for all the detailed design parameters. In current industry practice, detailed simulation models and optimization algorithms are widely used for this purpose. Though design decisions made during the early stages have a far greater impact on the final design quality and cost, few decision support tools are available for conceptual design.

During conceptual design, large areas of the design space are explored and candidate designs evaluated with respect to multiple criteria. Which tradeoffs are important at any stage during design, however, are not known *a priori* and depend on the results of other tradeoff analyses. Additionally, as cost is often a crucial criterion in system design, multiple attribute tradeoffs with combined performance and cost models are essential. Furthermore, models used for conceptual design tradeoff analyses are often based on incomplete and quickly evolving system descriptions. Therefore, the critical capabilities for a conceptual design system are flexibility in defining and performing tradeoff studies, ease of integrating models from multiple perspectives, and support for quick development and incremental modification of models.

Several research efforts are aimed at developing collaborative design environments that integrate multiple models and tools used by engineers (e.g., Herman and Lu, 1992; Cutkosky, 1993; McGuire et al., 1993; Sriram and Logcher, 1993). However, these environments are only appropriate for the detailed stages of design. Conceptual design research, on the other hand, has mainly focused on the role of optimization and robust design (e.g., Choy and Agogino, 1992; Dixon, Orelup and Welch, 1993), without any emphasis on how to use integrated models. Currently, conceptual design decisions are often made without recourse to integrated models. In some cases, spreadsheets and specialized FORTRAN programs are used, but these

tools are not flexible because the knowledge of what tradeoff studies are needed is hard-coded into them and it is difficult to change or refine these models after they are first created.

In order to overcome the difficulties discussed above, a conceptual design methodology should use a declarative representation for design models. Algebraic equations are typically the natural representation for conceptual design analysis knowledge. Such a representation separates the models from the mechanisms that control their use, thus making it easy to develop models incrementally and integrate them easily with other models. Using declarative knowledge for tradeoff analysis requires the solution of systems of non-linear equations. A flexible and rapid tradeoff study capability further requires a method to automatically produce control code from simple specifications. Constraint management techniques provide a powerful mechanism for accomplishing these tasks. They represent the equations as constraints between variables, and propagate the changes in variable values across the constraint network. In doing this, they generate computational plans necessary for solving the systems of equations as well as performing tradeoff analysis.

The robustness of numerical methods for solving systems of non-linear equations deteriorates rapidly with the number of equations that need to be solved simultaneously. The main strength of constraint management approaches is in decomposing large systems of equations into subsets of more manageable size, which can be solved individually before being combined to obtain the overall solution. Constraint management approaches have, therefore, been used by several researchers in creating design systems. Bouchard, et al. (1988) use directed constraints between design variables and numerical solution approaches for rapid production of trade-off studies. This approach, however, forces the designer to decide *a priori* which variables are input and which are output. Serrano (1987) has developed a constraint management approach based on bipartite matching for efficiently decomposing large systems of algebraic equations, and strong component identification for determining subsets of equations that need to be solved simultaneously. Ramaswamy and Ulrich (1993) have extended this work by developing a spreadsheet interface to the constraint system. Fromont and Sriram (1992) use planning techniques to add flexibility to such systems, such as allowing constraints to be added incrementally. Ward (1989) has extended constraint management for propagating interval values. However, this is only practical for linear systems.

In these approaches, the simultaneous subsets of equations are solved either symbolically or numerically, without further decomposition. In practical applications, where such subsets can include several tens of simultaneous nonlinear equations, robust solution techniques are not available. Krishnan, et al. (1990) discuss issues in user-directed constraints as well as suggestions for further decomposing strong components representing the equations that need to be solved simultaneously. However, their treatment is not comprehensive in either aspect.

We have developed a constraint management methodology and implemented a prototype system, called Design Sheet, for conceptual design tradeoff studies. It uses a graph-theoretic decomposition process, which not only identifies subsets of equations that need to be solved simultaneously, but further decomposes such subsets to improve the robustness of the overall solution procedure. This is one of the main reasons why Design Sheet scales up for solving practical analysis problems, which require from a few hundreds to a few thousands of equations to be solved. The system interface is designed to allow the user to easily develop the model, modify the independent variables, change variables values, and specify tradeoff studies.

DESIGN SHEET – A CONCEPTUAL DESIGN SYSTEM FOR TRADEOFF STUDIES

Design Sheet is a conceptual design system for facilitating tradeoff studies. It integrates constraint management techniques, symbolic mathematics and robust equation solving capabilities with a flexible environment for developing models and specifying tradeoff studies. It uses principally algebraic equations to represent a model[1], from which it automatically derives computational procedures required for solving systems of non-linear equations based on user-specified tradeoff studies.

Design Sheet Methodology

In order to effectively deal with large systems of algebraic equations, Design Sheet decomposes the systems into smaller, more manageable parts. As an example, consider the system of equations shown in Figure 1. Also, suppose that the variable g is an independent variable, whose value is specified by the user. Even though this is a relatively small system of equations, solving this system simultaneously for all 6 variables is quite hard and may not be fruitful. This is because non-linear equation solving methods have increasingly poor convergence properties as the number of equations and variables grows.

1. $a^2 + \log(c) = d$
2. $d \sin(f) + g = 0$
3. $a - c = e^2$
4. $b = e + f$
5. $d^2 + 5dg = 6$
6. $ac = e - f$

Figure 1. A system of non-linear equations.

A better solution strategy can be found by rearranging the equations. Figure 2 shows the rearranged equations in the form of a matrix, with equations and variables corresponding to rows and columns, respectively. A box in the matrix is colored if the variable in that column is present in the equation for that row. From the matrix, we can see that it is possible to first solve

[1] Besides simple algebraic expressions, Design Sheet allows interpolation and look-up tables, conditional expressions and external functions and subroutines in Lisp or FORTRAN.

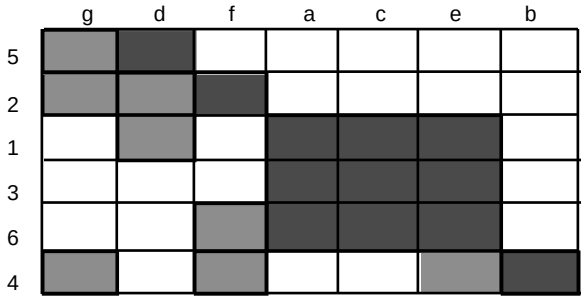

Figure 2. Reordering the system of equations.

equation 5 for d, then equation 2 for f. Once we have the values for d and f, equations 1, 3, and 6 can be solved simultaneously for a, c, and e. Finally, given the values of e, f, and g, equation 4 can be solved for b. The computational plan for solving the entire system is shown in Figure 3. Essentially, this decomposition has simplified the problem of solving a 6x6 system of equations down to a problem of solving a 3x3 system and three problems of solving a 1x1 system.

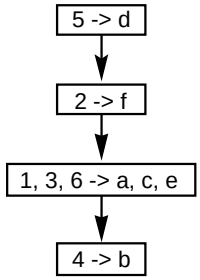

Figure 3. A computation plan.

In order to do this kind of decomposition, Design Sheet represents equations and variables in a bipartite graph, and makes use of several graph search algorithms to direct, group and reduce the graph. The bipartite graph is made up of two kinds of nodes, an equation node for each equation and a variable node for each variable. Edges in the graph connect equation nodes to variable nodes, and indicate that the variable is present in the equation. The bipartite graph for the system of equations in Figure 1 is shown in Figure 4.

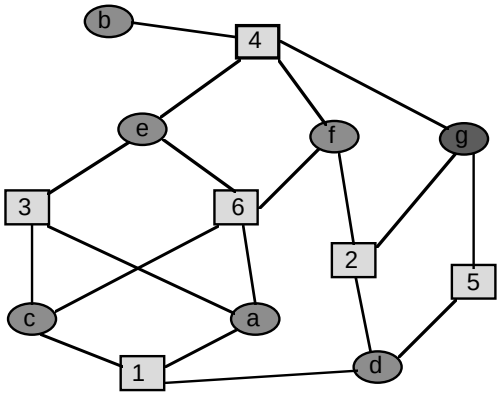

Figure 4. Bipartite graph of the 6x6 system.

The decomposition process involves several steps – directing the graph, variable determination, plan construction and component decomposition. These phases are briefly discussed here. More detailed descriptions can be found in Buckley, Fertig and Smith (1992) and Smith (1995).

Directing the Graph. The first step in decomposition is to assign directions to many of the edges in the graph. Directing the graph amounts to finding a consistent way of using all of the equations, so that no two equations would be used to compute the same variable. Directing is done according to the following rules:

1) If a variable is independent, all edges are directed away from the variable.
2) For each equation, there is exactly one edge directed away from the equation.
3) If there is an edge directed into a variable, all other edges are directed away from that variable.

Rule 1 corresponds to the fact that an independent variable will always be used as input in all equations where it appears. Rule 2 corresponds to the fact that an equation can only be used to compute one variable at a time. Rule 3 corresponds to the fact that once a variable is computed, it is used as input in all other equations where it appears.

Figure 5 shows one of four possibilities for directing the graph from Figure 4. In the current example, all the edges are directed by the above rules. However, this would not be the case if the graph is under-constrained, which arises when there are more variables than equations.

Directing is accomplished using a variant of the Ford-Fulkerson algorithm for finding maximal matchings on bipartite graphs (Cormen, Leiserson and Rivest, 1991). This algorithm uses a two phase approach. First, it finds an initial (not necessarily maximal) pairing of equation and variable nodes. Then, it finds paths in the graph where directions can be reversed to improve the matching. In reality, Design Sheet builds up the bipartite graph and does the directing incrementally. When a new equation is added, or a variable is declared independent, first the graph and then its edge directions are modified to reflect the new constraints. As a

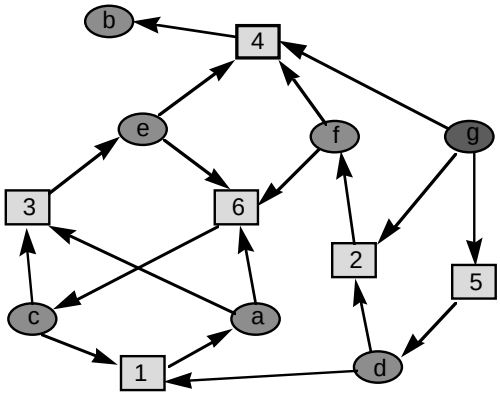

Figure 5. The directed bipartite graph.

result, Design Sheet always has an initial matching at hand, and only uses the path reversal phase of the matching algorithm.

Variable Determination. Once graph directing is complete, the second step in the decomposition process is to decide which variables are determined by the current set of independent variables. Much of this can be done with the aid of simple constraint propagation rules:

1) If a variable is independent, it is determined.
2) If all edges of an equation are directed and all immediate predecessors of the equation are determined, the successor of the equation is determined.

In the current example, we can use the first rule to conclude that the independent variable g is determined. The second rule is then used to determine that the variables d and then f are determined. However, we are stuck when we reach equations 1, 4 and 6, because their predecessors are not all determined.

In the directed graph from Figure 5, there are cycles among the variables a, c, and e, and the equations 1, 3, and 6. These nodes and the directed edges between them, form what is called a strongly connected component (Cormen, Leiserson and Rivest, 1991). A strongly-connected component (SCC) is a maximal set of nodes such that there is a directed path from any node in the set to any other node in the set. A SCC corresponds to a set of equations that can be solved simultaneously, if all variable predecessors to the component are determined. Given this definition, the third propagation rule can be stated as:

3) If all edges connected to equation nodes in a component are directed, and all predecessors (not already in the component) of the component equation nodes are determined, the variables in the component are determined.

In the current example, all edges connected to equation nodes in the component are directed. The external predecessors to the equation nodes in the component are the variables d and f, which are both determined. We can, therefore, conclude that the variables a, c, and e are determined. Once this is done, rule 2 can be used again to conclude that the variable b is determined. In general, it is necessary to repeatedly apply rules 2 and 3 until no further propagation is possible.

To apply rule 3, we must identify SCCs in the graph. This is accomplished using a standard backward search and marking algorithm (Cormen, Leiserson and Rivest, 1991). In the worst case, this algorithm takes time linear in the number of nodes and edges in the graph. We can further limit the time spent searching for SCCs by only examining the fringe nodes where simple propagation (by rules 2 and 3) gets stuck. While there may be other SCCs in the partially directed graph, rule 3 only applies when all the predecessors of the component are determined. This will only happen when simple propagation bumps up against the component.

Plan Construction. Once Design Sheet has figured out which variables are determined in the graph it can construct a plan for computing the value of any or all of the determined variables. If I is the set of variables of interest, then we only care about that subset of the graph consisting of all nodes that have directed paths to nodes in I. These are the variables and equations needed to compute the values for the set I. In this process, for each component, we coalesce all equation nodes into a single composite component equation node, and coalesce all component variable nodes into a single component variable node. This gets rid of all cycles within the graph. A plan for computing the values of the variables I now corresponds to any total ordering of the remaining directed acyclic graph (DAG).

In the current example, if the variable of interest is b, the resulting DAG is shown in Figure 6. The only linearization of this DAG is [g, 5, d, 2, f, {1,3,6}, {a,c,e}, 4, b]. This corresponds to the plan given earlier in Figure 3.

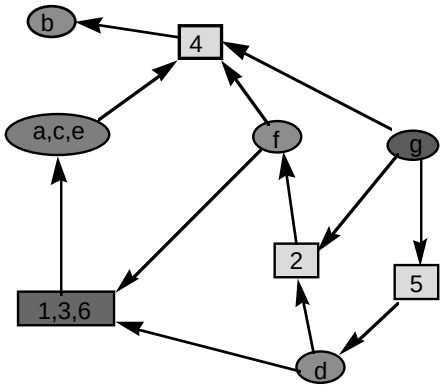

Figure 6. Collapsing the component.

Component Decomposition. In the example above, Design Sheet was able to decompose a 6x6 system of equations and variables down into three 1x1 systems and a 3x3 system. However, it must still solve the 3x3 system. We could guess initial values for the three variables, use the equations to compute errors and iterate. However, as we noted earlier, the robustness of this process deteriorates rapidly as the number of equations and variables increases. By judiciously choosing the iteration variables we can often do much better. For example, suppose that we guess an initial value for variable a. Using this guess in equation 1, we can compute a value for c. Using these values for a and c in equation 3, we can compute a value for e. Finally, we can use equation 6 to compute an error for a. We have reduced the problem to one involving only a single iteration variable with some intermediate propagation.

All variables in a SCC do not work equally well for this purpose. For example, if we choose e as an iteration variable, no immediate propagation is possible and we are forced to guess at a second variable. To improve the speed and robustness of numerical iteration, Design Sheet searches to find a minimal set of iteration variables for each SCC. It uses a heuristically guided branch and bound search together with smart pruning of the set of candidate iteration variables. Loosely, the algorithm functions as follows:

1) Select a variable in the SCC that is not yet determined and treat it as independent.

2) Find all simple consequences of this choice (i.e., find all other variables in the SCC that are now determined by simple propagation).
3) If any variables in the SCC remain undetermined, repeat the process.

When a solution is found, the system backtracks and tries other variable choices in step 1, in an effort to find a smaller iteration set. If the best set found so far is of size k, the algorithm will backtrack as soon as the current candidate solution grows beyond size k-1. The algorithm has another important improvement. Suppose a variable v is chosen in step 1 and another variable v' is determined from it (and other variables already in the iteration set). Since v allows the determination of v', v is at least as good a choice as v'. As a consequence, v' is not considered as a candidate for step 1. A more detailed description of this algorithm can be found in Smith (1995).

In general, the problem of finding a minimum size iteration set is NP-complete (Smith, 1995). This means that the above search may take time that is exponential in the minimum number of iteration variables required. Typically, this has not been a problem for Design Sheet, even in practical applications with SCCs involving as many as 150 equations. These SCCs are loosely coupled and can usually be solved using only 2 or 3 iteration variables. In such cases, the search algorithm finds a minimum size iteration set within a few seconds.

Difficult and Forced Directions. With non-linear systems, it is sometimes much easier to use an equation in one direction than another. Other times, an equation cannot be solved in a particular direction. Figure 7 shows two such equations and the corresponding parts of the constraint graphs. Consider the first equation in Figure 7. Given y and z, it is easy to use this equation to compute x. Similarly, given x and z it is easy to use this equation to compute y. In contrast, computing z from x and y requires numerical iteration. Design Sheet accounts for this by penalizing the assignment of certain edge directions. Another special situation arises in the case of the second equation in Figure 7. It is not possible to solve this equation for x. Design Sheet accounts for this by specifying that certain edge directions are required or forced. The graph representation for this case is also shown in Figure 7.

Direction penalties and forced directions add another level of complexity to all of the graph algorithms discussed above. For example, if direction penalties exist within a SCC, the component decomposition algorithm must consider how the cost would differ if simple propagation were done in a different order, resulting in different error equations. A more detailed description of these nuances can be found in Smith (1995).

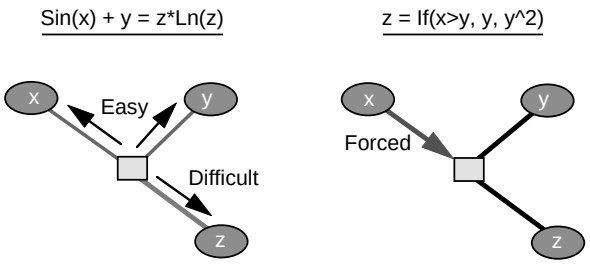

Figure 7. Difficult and forced directions.

Capabilities of Design Sheet

Important activities during conceptual design include "what-if analysis" to determine the consequences of individual design changes on system performance and "tradeoff studies" to explore the tradeoffs between competing objectives over large areas of design space. Design Sheet provides a spreadsheet like interface for what-if analysis, and a menu-driven graphical interface for performing design tradeoff studies.

What-if Analysis. The spreadsheet interface allows the designer to type in new values for different independent or input variables. Design Sheet propagates these values across the constraint network to update the values of the affected dependent or output variables. This allows the designer to study the implications of alternative designs, before deciding on final design choices. In contrast to a typical spreadsheet, Design Sheet allows the user to change which variables are input (or independent) variables. When the user changes the choice of inputs at run time, Design Sheet automatically inverts the model and produces the computational procedures needed to solve the resulting equations. Of course, Design Sheet can only invert where it is mathematically feasible to do so.

Design Sheet can also use external programs (Lisp or FORTRAN) to define relationships between parameters, and integrate them with other parts of the model made up of purely mathematical equations. It can also reverse the flow through such programs transparently. Of course, because it does not have information about the internals of an external program, it can only use it as a black box. Inverting flow through such programs will therefore be an iterative process.

Tradeoff Studies. The main purpose of Design Sheet is to perform tradeoff studies, which involve determining the effect of different values of independent variables on the values of dependent variables. In Design Sheet, the user defines new tradeoff studies simply by specifying the independent and dependent parameters of interest and the ranges in which to vary the independent variables. Design Sheet provides two interfaces for displaying results of tradeoff studies, namely trade tables and plots. Trade tables are mini spreadsheets, which allow specialized trades to be done separate from the main model; Design Sheet automatically creates such a spreadsheet by copying only the relevant part of the model, based on user-specified variables of interest.

Design Sheet also provides a flexible interface for plotting the results of tradeoff studies using either two-dimensional or cross plots. Two-dimensional plots are used to plot the effect of an independent variable on a dependent variable, at possibly discrete values of other independent parameters. Cross plots

are used to plot two dependent parameters, as several independent parameters are varied over specified ranges. The latter are especially powerful for studying performance-cost and other multiple-objective tradeoffs, and for quickly exploring large areas of design space.

Error Propagation. During conceptual phases of design, the designer is often forced to make estimates for the values of certain independent parameters, because accurate information is not available early in the design process. However, the designer would still like to know how errors in these estimates affect the various tradeoffs. This capability is especially useful when integrated product and process models are used in design. Manufacturing parameters such as yields are better represented as statistical distributions, and any design decisions should consider the effects of these distributions on down-stream performance attributes. Though Design Sheet does not provide facilities for modeling statistical distributions, it allows the designer to specify standard deviations for the independent variables. Design Sheet uses the same constraint network that it uses to solve the system of equations for propagating the effects of errors in independent variable estimates. It assumes that the errors in the different independent parameters are uncorrelated and performs a first order error analysis.

Constrained Optimization. Optimization is an important part of design process. The designer may wish to maximize the performance subject to cost constraints or minimize the cost to meet certain performance targets. Design Sheet has a limited capability for constrained optimization. It allows any dependent variable to be either minimized (or maximized), subject to user-defined inequality constraints. The basic algorithm determines if a given constraint is active, and if so adds it as an equality constraint, before solving the minimization problem. It uses a modified gradient-search algorithm for minimization. Design Sheet also provides the capability to produce contour plots for an objective parameter with respect to independent parameters, with superimposed inequality constraints.

DEMONSTRATION OF DESIGN SHEET

Design Sheet has been successfully used on practical system analysis problems, for example, to study integrated performance and cost tradeoffs of aircraft and spacecraft concepts. These analyses involved more than a thousand non-linear equations form several disciplines. An order of magnitude increase in the number of tradeoff studies has been realized. More importantly, Design Sheet has enabled new ways of visualizing and exploring the design space, because it allows parameters (e.g., cost) that are normally thought of as outputs of a model to be treated as inputs that can be freely varied. The results from these applications have not been published owing to the competition-sensitive nature of this information. Design Sheet is further being used to develop an "Infrared Seeker Performance Model" under the DARPA MADE program to facilitate tradeoff studies during the conceptual design of thermal imaging systems. This section demonstrates the capabilities of Design Sheet, especially the ease with which new criteria can be added to the model and novel tradeoff studies obtained without any reprogramming.

Designers at Rockwell Autonetics and Missile Systems Division currently use FLIR92 (NVESD, 1993), a computer model developed by the U.S. Army Night Vision and Electronic Sensors Directorate, for system level design analysis of thermal imaging systems for missile seekers. This model is quite accurate and has proved very useful in the early-stage analysis of alternative seeker designs. However, the lack of integrated trade-study capability limits the number of alternatives that are evaluated, and does not lend the FLIR92 model for easily searching large areas of design space before making crucial design decisions. A further limitation of this model is its inflexibility in dealing with design criteria not already programmed into the model.

The Design Sheet based seeker model seeks to integrate criteria from optical, mechanical and operational aspects of seeker design into a single model, so that performance trades from multiple perspectives can be effectively used during seeker design. As a first step in developing the overall model, the core module for predicting the thermal imaging system performance, based on the FLIR92 model, has been developed in Design Sheet. The model includes both scenario specific (atmospheric, terrain, target, etc.) and design specific (optics, electronics, sensor, software, display, etc.) information.

Thermal Imaging Systems Model – FLIR92

The FLIR92 thermal imaging systems performance model uses basic system-level design and operational parameters to predict whether a system meets performance requirements necessary to meet a target acquisition and discrimination task. The model calculates widely accepted system performance measures, namely the modulation transfer function (MTF), the minimum resolvable temperature difference (MRTD), and the minimum detectable temperature difference (MDTD), for both scanning and staring thermal imaging systems.

Illustrated in Figure 8 are the major system components used in the model. Radiation from the target and the background scene is captured by the seeker optics. The photons are passed through the lens and focused onto a sensor which converts the energy into an electric signal. The system electronics process the signal from the detector into a format which is suitable for re-display to a human observer. The observer then makes a decision as to whether the viewed target is discernible based on the displayed information.

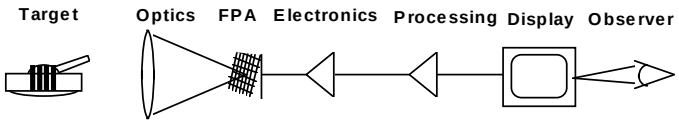

Figure 8. Thermal imager block diagram.

The model consists of equations which reflect how the signal is modulated from the target to the observer. The model calculates the total system transfer function using the linear filter theory, by calculating MTFs for individual system components and then multiplying them together. The model calculates MTFs due to system components such as optics and electronics, sensor design characteristics such as scene phasing, and operational environment effects such as sensor motion. The observer is also modeled as a system component for the purpose of calculating the MTF. The component MTFs are divided into pre-filters and post-filters, depending on whether they occur before or after sampling, and the product of pre-filter and post-filter MTFs yield the total system MTF. Noise filters are used to model component MTFs after noise injection. An example MTF for the diffraction-limited optics (H_{odl}) is given below in terms of the spatial frequency (f_s), diffraction wavelength (λ) and optics aperture diameter (D_o).

$$H_{odl}(f_s) = \frac{2}{\pi}\left(Cos^{-1}\left(\frac{\lambda f_s}{D_o}\right) - \left(\frac{\lambda f_s}{D_o}\right)\left(1-\left(\frac{\lambda f_s}{D_o}\right)^2\right)^{1/2}\right)$$

MRTD and MDTD are used as system performance measures. The MRTD is the minimum temperature difference between a standard bar target (four 7:1 aspect ratio bars) and the background that is required in order for a standard observer to just fully resolve the target. MRTD is defined for the horizontal and vertical directions, depending on the target pattern orientation. MRTD depends on the system transfer function (resolution) and the system sensitivity. The MDTD is the minimum temperature difference between a square target and the background required to make the target detectable to a standard observer. MRTD is the best overall indicator of thermal imager performance. MDTD is a less stringent performance measure, but is useful in specialized applications where point source detection is required, such as targets against uniform backgrounds.

The model also has equations for the effects of various noise parameters on system performance (MRTD and MDTD). The noise analysis methodology uses the concept of directional averaging to isolate system noise into eight components, corresponding to both spatial and temporal aspects of pixel, row, column and frame effects. The total noise is given by the root sum square of the components. The random spatio-temporal noise component is modeled as a function of the detector detectivity and the temperature. The other components are generally represented by scaling factors which multiply the random spatio-temporal noise, but can be independently input to the model. A detailed description of the model can be found in NVESD (1993).

Basic Characteristics of the Design Sheet Thermal Imaging Systems Model

A Design Sheet model to duplicate the functionality of the FLIR92 model has been completed, and will serve as the core module of the overall seeker performance model. Currently, the Design Sheet model has 17 component MTFs and eight noise components, represented by 124 scalar variables, 72 constraints and 69 transforms. They model both the staring and scanning type of infrared imagers operating in the mid-wave to long-wave infrared spectral bands. In addition to the equations for MRTD and MDTD, the model has equations for predicting the probability of detection of a target based on system MRTD, target size and target range. It is this ability of Design Sheet to easily integrate additional model fragments that contribute to the flexibility of the resulting seeker design system.

The constraint network depicting the dependencies between the equations and variables in the model is shown in Figure 9. As the total constraint network is too large to be displayed here, Figure 9 only shows the part of the network that is relevant to the tradeoff studies described in this paper. The variables of importance are the design parameters, focal length and f/#; the performance parameters, field of view (FOV), probability of detection (POD) and minimum resolvable temperature difference (MRTD); and the operational characteristics, target range and spatial frequency (fspat). Focal length is the effective focal length and f/# is the ratio of the focal length to the aperture diameter of the overall optical system.

The arrows in Figure 9 denote the default direction of data flow in the constraint network. In this mode, given the design and operational parameters, the model calculates the performance metrics, and is the mode in which the original FLIR92 model is used. Test cases have been run in this mode to verify that the results produced by the Design Sheet model are accurate with reference to the original FLIR92 model. In order to calculate the performance metrics, the user interacts with the model through the top-level graphical user interface shown in Figure 10. There are four panes, the right hand side is the variable pane, and the left hand side is divided into the command pane, the display pane and the relation pane, respectively from top to bottom. The variable pane serves as a

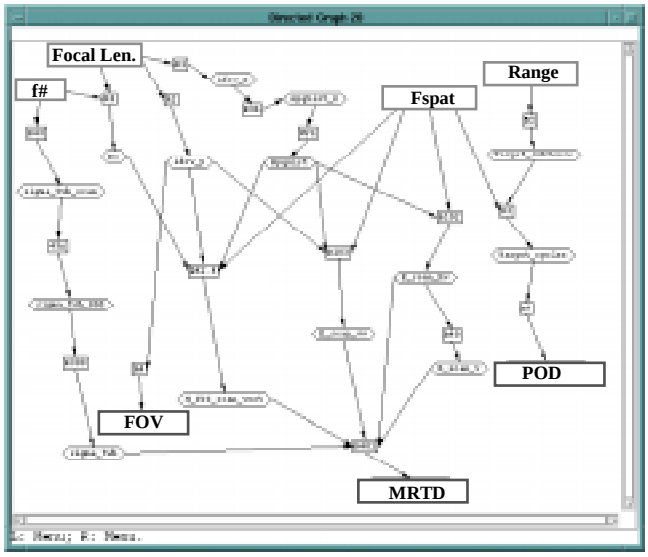

Figure 9. A partial constraint network for the seeker model.

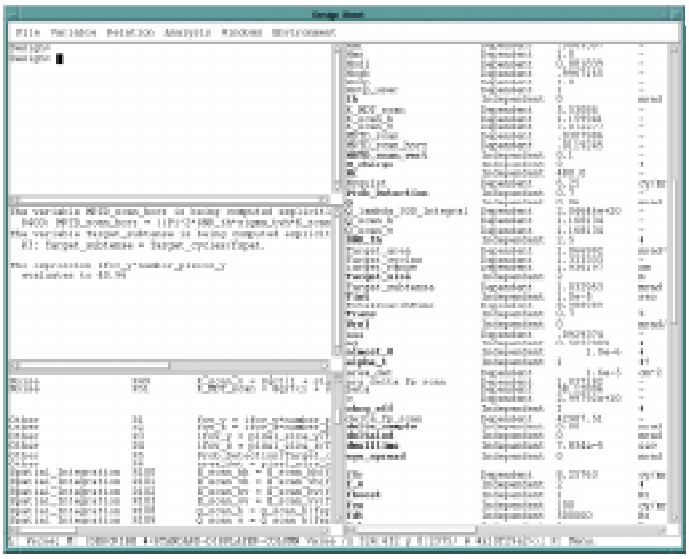

Figure 10. Design Sheet top-level user interface.

spreadsheet, and when values are changed for independent variables, they are propagated across the constraint network and new values calculated for the performance attributes. The menu bar of the interface can be used to access various commands, the most interesting of which are the commands for defining tradeoff studies and displaying the results.

In addition to calculating the MRTD for a specific seeker design, the seeker performance model can also be used in the default forward mode to calculate the various modulation transfer functions. Figure 11 shows the plots of the total system MTF and some of the component MTFs as a function of spatial frequency. Such a tradeoff is useful in identifying critical components for further analysis. Producing this graph requires only minimal effort, as the plotting capability is integral within Design Sheet. All that is required of the user is to type in the x-axis and y-axis variable names, and the ranges for the x-axis variable. The integrated tradeoff capabilities have proved to be very important in promoting user acceptance of Design Sheet.

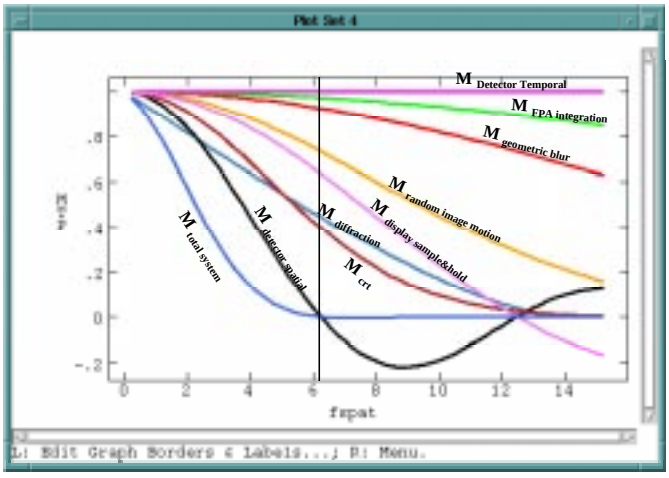

Figure 11. System MTF and some component MTFs.

Using the Design Sheet Model for Tradeoff Studies

The ability to study the performance characteristics of a specific seeker design is an important capability in itself. However, the uniqueness of Design Sheet is in being able to use this model for performing tradeoff studies over large tracts of design space, before selecting a specific design. Especially interesting and extremely useful are the tradeoffs that involve fixing some of the performance requirements, such as the tradeoff between target range and field of view for a specified probability of detection. Some interesting tradeoff studies are reported in this section. Though the model itself has equations for both scanning and staring systems, the tradeoff studies shown here are only for the scanning system.

An important characteristic that is often considered by seeker designers is the tradeoff between MRTD and spatial frequency. A good seeker design should not only have smaller values of MRTD, but the variation of MRTD with spatial frequency should be small for spatial frequencies of interest. In order to study the effect of different optical system designs on the tradeoff between MRTD and spatial frequency, the designer can generate a plot of this characteristic for different values of optical system parameters. Such a tradeoff between horizontal MRTD and spatial frequency, for a focal length of 250 mm and varying values of f/#, can be generated quickly and easily in Design Sheet, and is shown in Figure 12. This tradeoff plot is obtained in a matter of seconds.

The MRTD is generally viewed as an output, while spatial frequency is the input, of the seeker performance model. In Design Sheet, however, the state of these two variables can be switched. This can be done easily and only requires a couple of mouse clicks; first the variable, fspat, is made undetermined and then the variable, MRTD, is made independent. The new state of the directed constraint graph is shown in Figure 13. In Figure 13, the independent parameters of interest, namely, f/#, focal length, range and MRTD, are circumscribed by blue rectangles, whereas the parameters calculated by the constraint network, namely, fspat, FOV and POD, are circumscribed by red rectangles. The new computational plan is automatically

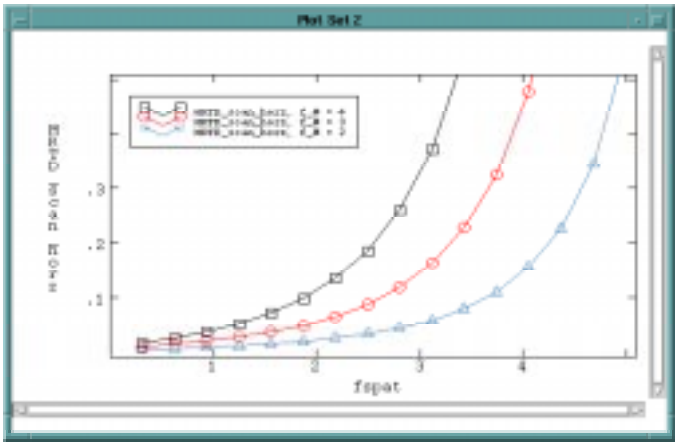

Figure 12. The effect of optics design on the tradeoff between MRTD and spatial frequency.

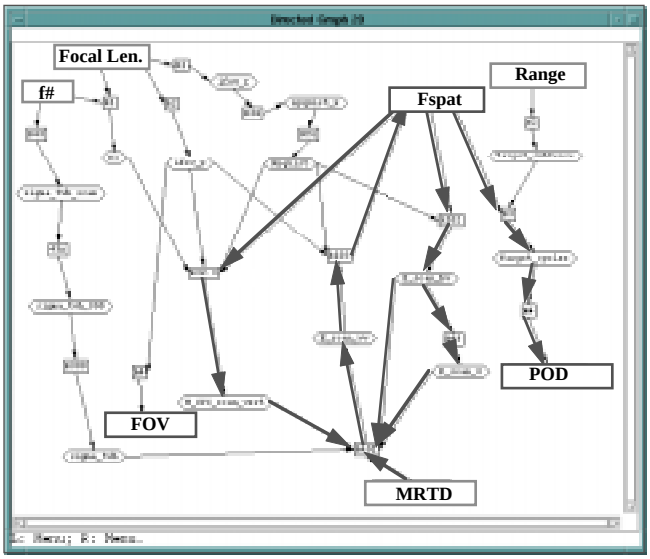

Figure 13. Constraint graph with MRTD as input.

determined by Design Sheet. The path of computation for probability of detection for a specified MRTD is shown by the highlighted (in red color) arrows. This plan requires a system of equations involving MRTD and spatial frequency to be simultaneously solved.

The model in this state can be used to specify a fixed value of MRTD, and a tradeoff between probability of detection (POD) and target range obtained. Such a tradeoff is very useful in the design of thermal imaging systems, because a minimum value of MRTD is often specified as a requirement. Figure 14 is a plot of this tradeoff, for MRTD of 0.1 and an effective focal length of 250 mm, for two different values of f/#. This tradeoff is especially useful during the conceptual stages, because the designer is able to tradeoff two performance attributes, namely POD and target range, for a strict requirement on MRTD. Figure 14 further shows how the optical system parameters can be adjusted to change the nature of this tradeoff.

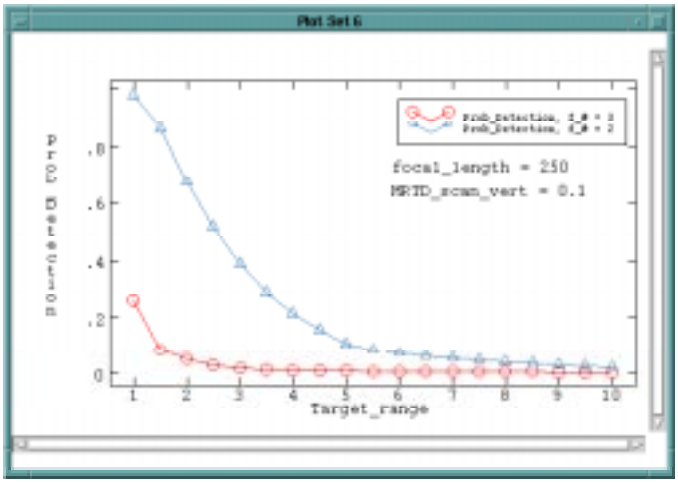

Figure 14. Tradeoff between target range and probability of detection for a specific value of MRTD.

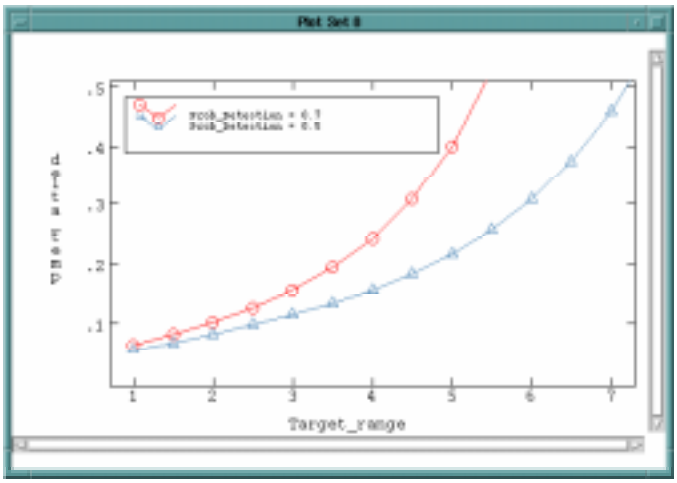

Figure 15. Tradeoff between MRTD and target range for a specified probability of detection.

If the critical requirement were probability of detection, instead of the MRTD, the flexibility of Design Sheet allows the designer to obtain a new tradeoff that is appropriate for this situation. The MRTD is now relaxed, and the probability of detection made as an input instead. Now, a tradeoff plot can be obtained between the MRTD and target range for a specified requirement on probability of detection. Such a plot for two different probabilities of detection is shown in Figure 15.

Finally, let us consider an even more interesting tradeoff study. The designer is often provided with requirements on the minimum temperature differences that need to be resolved (MRTD) as well as the probability of detection (POD) that is desired. Under these circumstances, there is a tradeoff between the target range and field of view given a specific MRTD and POD. Knowing this tradeoff would enable the conceptual designer to provide a more appropriate specification on the target range in the requirements specification stage.

Making both the MRTD and the POD independent variables requires both spatial frequency and target range to be relaxed as inputs. Figure 16 shows the directed constraint graph for this state of the model. The data flow in the model from focal length, f/#, MRTD and POD to field of view (FOV) and target range is in a direction that is reverse of the data flow in the traditional use of the model. It is this ability to reverse the flow directions that makes Design Sheet a powerful tool for performing tradeoff studies in support of conceptual design.

Figure 17 shows the tradeoff plot between field of view (in the vertical direction) and target range for specified values of 0.1 for the MRTD (in the vertical direction) and 0.7 for the probability of detection. This plot is called a cross plot and plots two dependent variables on the x and y axes. The independent variables that are varied are the focal length and f/#, and the different values of these variables are shown on the plot. Based on the trade-off shown in Figure 17, and other relevant information about the operational regimes for field of view (FOV) and target range, the designer can choose appropriate values for focal length and f/#.

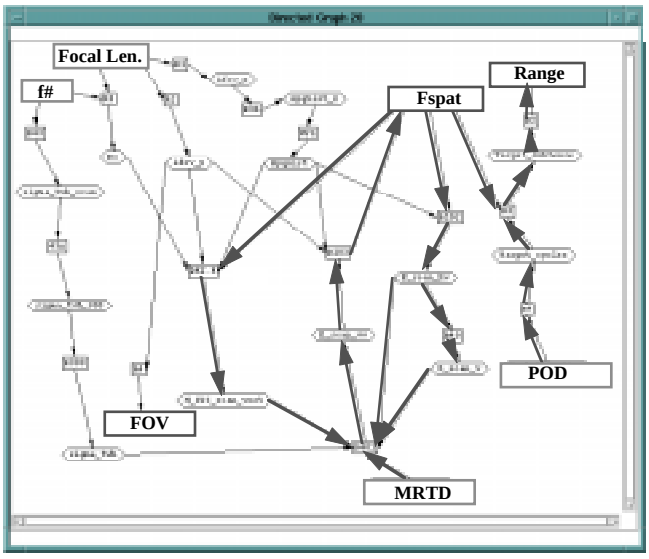

Figure 16. Constraint graph with both MRTD and POD being independent.

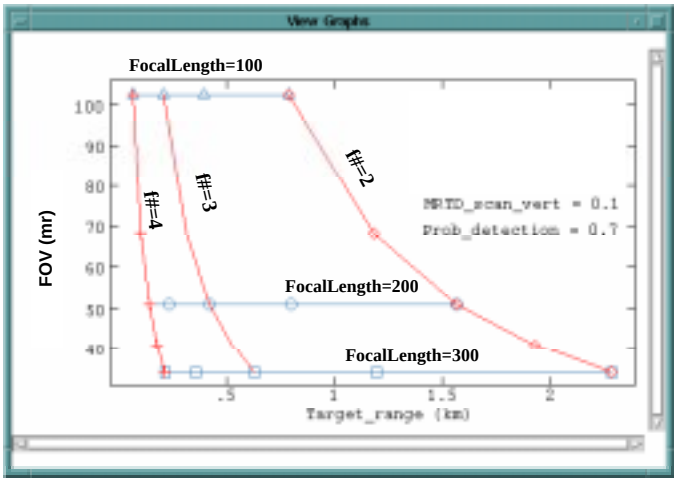

Figure 17. Tradeoff between field of view and target range.

The tradeoff studies shown in this section are only examples and are by no means the only ones used by designers in developing infrared seekers. Design Sheet does not predefine the tradeoffs that are needed by designers, but makes the generation of new tradeoffs quite easy.

CONCLUSIONS

The integration of constraint management techniques with a flexible environment for tradeoff studies has resulted in a powerful tool for conceptual design. By allowing models to be represented in the form of mathematical equations, separate from any control code, Design Sheet permits models to be developed incrementally and integrated with other models easily. Design Sheet automatically derives control code necessary for performing tradeoff studies based on simple user specifications, thus allowing the designer to define new tradeoff studies easily. This along with the ability to determine which variables are input at run time make Design Sheet a flexible tool for conceptual design and requirements analysis.

The success of Design Sheet on practical applications is a result of the graph theoretic algorithms used for constraint management, which scale gracefully to extremely large sets of equations. Like other constraint-based systems, Design Sheet decomposes the system of equations into subsets that need to be solved simultaneously. However, in contrast to other systems, Design Sheet further decomposes these simultaneous subsets before applying numerical solution techniques. This is crucial because the robustness of numerical approaches for solving systems of non-linear equations deteriorates rapidly with the number of equations. The thermal imaging systems design example clearly demonstrates not only that complex models can be implemented using Design Sheet but also the ease with which tradeoff studies can be performed. Tradeoff studies shown here have been obtained in a matter of seconds to minutes, whereas they could have taken hours if not days to produce using traditional means.

Design Sheet has proved very useful on practical conceptual design analysis problems, where the model can be represented as a network of constraints between design and performance attributes. Often, during conceptual design, however, design alternatives cannot all be modeled using a single constraint network. Design Sheet does not offer automatic support for comparing design alternatives requiring different constraint networks. Another current limitation of Design Sheet is the inadequate support provided for including differential equations as part of the constraint network. A future direction for this research is to investigate adding more principled support for solving ordinary differential equations. An approach we are considering will extend the constraint management algorithms to handle function-valued attributes.

ACKNOWLEDGMENTS

This research has been funded in part by DARPA MADE program administered through the United States Air Force Manufacturing Technology Directorate under contract F33615-94-C-4426. We wish to thank Anne Hemingway of Rockwell Autonetics and Missile Systems Division for help with the modeling of the infrared seeker systems and the interpretation of the tradeoffs.

REFERENCES

Bouchard, E. E., Kidwall, G. H., and Rogan, J. E., 1988, "The Application of Artificial Intelligence Technology to Aeronautical System Design," AIAA 88-4426, AIAA Aircraft Design Systems and Operations Meeting, Atlanta, Georgia.

Buckley, M. J., Fertig, K. W., and Smith, D. E., 1992, "Design Sheet: An Environment for Facilitating Flexible Trade Studies During Conceptual Design," AIAA 92-1191, Aerospace Design Conference, Irvine, California.

Choy, J. K., and Agogino, A. M., 1992, “SYMON: Automated Symbolic Monotonicity Analysis System for Qualitative Design Optimization,” *Proceedings of the ASME International Computers in Engineering Conference*, Chicago, Illinois, pp. 305-310.

Cormen, T., Leiserson, C., and Rivest, R., 1991, *Introduction to Algorithms*., McGraw-Hill Book Company, New York.

Cutkosky, M., R., 1993, “PACT: An Experiment in Integrating Concurrent Engineering Systems,” *IEEE Computer*, Vol. 26, pp. 28-37.

Dixon, J. R., Orelup, M. F., and Welch, R. V., 1993, “A Research Progress Report: Robust Parametric Designs And Conceptual Design Models,” *Proceedings of the 1993 NSF Design and Manufacturing Systems Conference*, SME, Dearborn, Michigan, pp. 499-506.

Herman, A. E., and Lu, S. C-Y., 1992, “Computer Methods for Distributed Reasoning to Support Concurrent Engineering,” *Proceedings of the Prolamat ‘92 Conference*, Tokyo, Japan, pp. 1-19.

Krishnan, V., Navinchandra, D., Rane, P., and Rinderle, J. R., 1990, “Constraint Reasoning and Planning in Concurrent Design,” Technical Report CMU-RI-TR-90-03, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

McGuire , J. G., Kuokka, D. R., Weber, J. C., Tenenbaum, J. M., Gruber, T. R., and Olsen, G. R., 1993, “SHADE: Technology for Knowledge-Based Collaborative Engineering,” *Concurrent Engineering Research and Applications*, Vol. 1, pp. 1-17.

NVESD, 1993, “FLIR92 Thermal Imaging Systems Performance Model,” Document 5008993, U.S. Army Night Vision and Electronic Sensors Directorate, Ft. Belvoir, Virginia.

Ramaswamy, R., and Ulrich, K., 1993, “A Designer’s Spreadsheet,” *Proceedings of the Design Theory and Methodology Conference*, DE-Vol. 53, ASME, New York, pp. 105-113.

Serrano, D., 1987, “Constraint Management in Conceptual Design,” Ph.D. Dissertation, MIT, Department of Mechanical Engineering, Cambridge, Massachusetts.

Smith, D., 1995, “Equation Decomposition in Design Sheet,” Technical Report, Rockwell Science Center, Palo Alto Laboratory, Palo Alto, California (in preparation).

Sriram, D., and Logcher, R., 1993, “The MIT DICE Project,” *Computer*, Vol. 26, pp. 64-65.

Ward, A. C., 1989, “A Theory of Quantitative Inference for Artificial Sets Applied to A Mechanical Design Compiler,” Ph.D. Dissertation, MIT, Department of Mechanical Engineering, Cambridge, Massachusetts.