10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 30 August – 1 September 2004, Albany, New York

Reconfigurability in MDO Problem Synthesis, Part 1

Natalia M. Alexandrov*

NASA Langley Research Center, Hampton, Virginia, 23681-2199, USA

Robert Michael Lewis†

College of William & Mary, Williamsburg, Virginia, 23187-8795, USA

Integrating autonomous disciplines into a problem amenable to solution presents a major challenge in realistic multidisciplinary design optimization (MDO). We propose a linguistic approach to MDO problem description, formulation, and solution we call reconfigurable multidisciplinary synthesis (REMS). With assistance from computer science techniques, REMS comprises an abstract language and a collection of processes that provide a means for dynamic reasoning about MDO problems in a range of contexts. The approach may be summarized as follows. Description of disciplinary data according to the rules of a grammar, followed by lexical analysis and compilation, yields basic computational components that can be assembled into various MDO problem formulations and solution algorithms, including hybrid strategies, with relative ease. The ability to re-use the computational components is due to the special structure of the MDO problem. The range of contexts for reasoning about MDO spans tasks from error checking and derivative computation to formulation and reformulation of optimization problem statements. In highly structured contexts, reconfigurability can mean a straightforward transformation among problem formulations with a single operation. We hope that REMS will enable experimentation with a variety of problem formulations in research environments, assist in the assembly of MDO test problems, and serve as a pre-processor in computational frameworks in production environments. This paper, Part 1 of two companion papers, discusses the fundamentals of REMS. Part 2 illustrates the methodology in more detail.

I. Introduction

For the purposes of our discussion in this and the companion paper, *Reconfigurability in MDO Problem Synthesis, Part 2*,[1] Multidisciplinary Design Optimization (MDO) refers to that part of the total design process that can be formulated as an optimization problem. The papers are concerned with an approach to reasoning about MDO problems that eases the handling of MDO problems during the formulation and computational solution processes.

MDO problems usually start out as collections of autonomous disciplinary analyses with diverse data formats. The autonomy, complexity, and diversity present a major challenge for the integration of disciplines into a nonlinear programming problem statement.

Numerical and analytical studies indicate that MDO problem formulation has direct influence on computational tractability of the resulting optimization problem.[2,3,4,5,6,7] However, in realistic MDO environments, it is often difficult to determine *a priori* whether a chosen problem formulation will produce

*Senior Research Scientist, Multidisciplinary Optimization Branch, Mail Stop 159. Senior Member AIAA.

†Assistant Professor, Department of Mathematics, P.O. Box 8795. Member AIAA. This research was supported by the National Aeronautics and Space Administration under NASA Grant NCC-1-02029 and by the National Science Foundation under Grant DMS-0215444.

satisfactory results. Reconfiguration may be required. The expense and complexity of problem integration usually leaves little or no room for experimentation with alternative formulations. Most commonly, the MDO problem is posed in a canonical form that ties an optimization code to the multidisciplinary analysis (MDA). Comparative properties of various problem formulations, including the effects of MDA and disciplinary autonomy, may be found in a number of papers.[8, 9, 5, 6, 7]

The difficulty is also pronounced, although not as expensive, in MDO test problems. Test problems, essential for algorithm development, mostly appear in the form of a particular formulation already implemented. Pulling such problems apart into subsystems that emulate disciplinary analyses is laborious and error prone. Attempting to disassemble test problems for numerical experimentation with MDO formulations gave us our original motivation to investigate formal approaches for reasoning about MDO that would minimize the effort in disciplinary and interdisciplinary specification.

Thus we see a clear need for flexible MDO problem implementation that should assist researchers and practitioners in formulating and reconfiguring MDO problems with ease, as well as in extracting information that would enable reasoning about various aspects of formulations and their use in the context of analysis and optimization.

It stands to reason that developing such a general, flexible methodology for managing disciplinary subsystems in MDO problem formulation and solution should be feasible. After all, the context of optimization requires a limited number of basic problem entities to be manipulated: design variables, objective and constraint functions obtained by evaluating the outputs of the contributing analyses, and possibly the associated derivatives. In fact, many recent developments in computational infrastructures (see Section II), have eased various implementation aspects for simulation-based design. The mechanics of abstract reasoning about MDO problem formulation and implementation, however, have remained elusive. One reason is, arguably, the difficulty of general specification of MDO components in sufficient detail when using algebraic notation or taxonomic notation, which are some of the most common abstractions for describing MDO. We faced this difficulty repeatedly in our earlier attempts to derive notation for MDO.

The present investigation suggests that the obstacles inherent in algebraic and taxonomic notations can be overcome by a linguistic, context-free-grammar based approach to MDO problem description, formulation, and solution. We call our approach *reconfigurable multidisciplinary synthesis* (REMS).

REMS is a conceptual framework that comprises an abstract language and a collection of processes that provide a means for dynamic reasoning about MDO problems in a range of contexts, with assistance from computer science techniques. REMS starts with a description of disciplinary data according to the rules of a grammar. Lexical analysis of the description, followed by linkages of multidisciplinary graphs, allow the researcher to manipulate basic computational components in a number of contexts. The components can be assembled into various MDO problem formulations and solution algorithms, including hybrid strategies, with relative ease. By relying on an abstract language, we have so far avoided the difficulties associated with the use of algebraic and taxonomic notations.

The range of contexts for reasoning about MDO spans tasks from error checking and derivative computation to formulation and reformulation of optimization problem statements. In highly structured contexts, reconfigurability can mean a straightforward transformation among problem formulations with a single operation. We hope that REMS will ease experimentation with problem formulations in research environments, assist in the assembly of MDO test problems, and serve as a pre-processor in computational frameworks in production environments. Although here we focus on MDO, we conjecture that REMS can be helpful in reasoning about complex systems in general and systems of systems in particular, not just in the context of formal optimization, but also in the contexts of various process simulations and decision making.

Part 1 of the two companion papers places this investigation in perspective with other efforts and describes the fundamentals of the approach, illustrating it with a simple example. Part 2 discusses some elements of the methodology in more detail.

II. Perspective in Relation to Other Methods

There are many connections between this work and other efforts in scientific and engineering computational infrastructure. We discuss a small sample of related work.

The general idea of computational components is pervasive in scientific computing, having to do with the object-oriented techniques used to abstract re-usable computational entities. Some component-based systems relevant to design are the Toolkit for Advanced Optimization[10] (TAO) and the Modeling Language for Mathematical Programming[11] (AMPL). These are component-based systems that connect function and constraint evaluation and the attendant derivatives to optimization algorithms in a systematic manner.

Methodologically, our work relies on abstractions of graph theory[12] and is similar in the basic approach to such well-developed areas as compiler construction[13] and automatic differentiation.[14] Wagner[15] used a different graph abstraction to examine various decompositions of MDO problems.

Etman *et al.*[16,17] also employ a compiler-like approach (χ language) to coordinate distributed design processes. We are investigating these techniques as well as other scheduling strategies[18] as potential models for representing time-dependent processes.

Many commercial and public-domain computational frameworks for design (ModelCenter[19] and DAKOTA[20] are examples) have been developed in recent years. Frameworks usually assume a particular MDO problem formulation and assist the user with entering the constituent analyses into the framework, to ease data transfers among disciplines and optimization algorithms by exposing the relevant variables and functions to the framework. Computational frameworks must also rely on various abstractions of computational components and be able to operate on these components.

Given the need for a formal reasoning technique and some suitable definition of problem components, the efforts mentioned here share some elements with REMS. Our goals are, however, complementary. The distinction may be summarized as follows. To our knowledge, most efforts start with a large, (at least conceptually) integrated system and then make decisions about suitable decomposition and coordination strategies. In REMS, we assume that the MDO problem exists as a set of independent systems, and our task is the synthesis of some MDO formulation from scratch. Our goal is to reason about the problem *before* it is integrated into a computational framework or is ready for coordination as a set of distributed processes. The reasoning is not only meant to identify reusable computational components but also to suggest the most suitable formulations and algorithms given the problem structure. We view frameworks and other coordination systems as targets for potential synergy with the present investigation: REMS is intended as a pre-processor for computational frameworks and coordination systems such as those described by Etman *et al.*[16,17]

Our other goal is likely common to most efforts in infrastructure development: to ease integration, no matter the problem formulation, and ease re-formulation if needed. Our approach to this goal is to minimize input required from disciplinary practitioners, to automate problem formulation to the maximum possible degree, and to provide automatic error checking at all stages of formulation. We believe this problem has to be attacked prior to the stage at which disciplines are currently entered into computational frameworks. This is another reason why we believe REMS may become promising as a potential pre-processor for frameworks.

III. Origins of Reconfigurability

The notion of reconfigurability[21,22] in the context of optimization is natural: any optimization algorithm supplies the system and the subsystems with design variable vectors and requires objective and constraint functions (and maybe their derivatives) from the system and subsystems. What exactly is required from the subsystems and when depends on the optimization problem formulation. In contexts with special structure, reconfigurability can be taken further than the simple reuse of components.

We illustrate the idea of reconfigurability on a two-discipline model problem and also point out the difficulty associated with algebraic notation in reasoning about general MDO problems. We consider three formulations: simultaneous analysis and design (SAND),[23] the straightforward fully integrated optimization

(FIO) formulation, and distributed analysis optimization (DAO).[9,6,7]

When viewed in a multidisciplinary context, each disciplinary analysis $A_i, i = 1, 2$, takes as its input a set of disciplinary design variables l_i, a set of shared or system-level design variables s, and some parameters that are outputs of the other discipline. The vector a_i represents the totality of outputs from a given discipline, while the parameters fed into discipline i are derived from the analysis outputs $a_j,\ j \neq i$, of the other discipline and are not directly manipulated by the disciplinary expert in discipline i. In algebraic notation,

$$a_i = A_i(s, l_i, T_i(a_j)),$$

where information is transferred from analysis i to analysis j, $j \neq i$, by the operator T_i.

The formulation treats the full multidisciplinary analysis (MDA) as explicit equality constraints. A simple version of SAND is

$$\begin{array}{cl} \underset{s,l_1,l_2,a_1,a_2,t_1,t_2}{\textbf{min}} & f(s, a_1, a_2) \\ \textbf{s.t.} & c_1(s, l_1, a_1) \geq 0 \\ & c_2(s, l_2, a_2) \geq 0 \\ & t_1 = T_1(a_1) \\ & t_2 = T_2(a_2) \\ & a_1 = A_1(s, l_1, t_2) \\ & a_2 = A_2(s, l_2, t_1). \end{array} \tag{1}$$

SAND ensures consistency among the shared design variables at the solution by introducing consistency constraints. Auxiliary variables t_1, and t_2 ensure consistency among the analysis outputs. Vectors c_i represent the disciplinary constraints.

This algebraic notation is very useful for describing problems with two disciplines in that it allows us to specify the functional dependencies in a problem formulation completely, which facilitates the computation of multidisciplinary derivatives. However, the notation is straightforward only for two disciplines. If we had three or more disciplines, the number of indices on the formulation entities would grow as would the function dependency lists. For instance, it would no longer suffice to say that s are shared variables: we would have to distinguish between variables shared among pairs of disciplines, where some may be duplicates. This would complicate the specification of minimal components for computing derivatives. The difficulty would be exacerbated as the number of disciplines grows in realistic problems or even in interesting test problems.

Returning to the notion reconfigurability, other MDO formulations may be viewed as derived from the SAND formulation by closing a particular set of constraints. In particular, a DAO formulation is

$$\begin{array}{cl} \underset{s,l_1,l_2,t_1,t_2}{\textbf{min}} & f(s, a_1(s, l_1, t_2), a_2(s, l_2, t_1)) \\ \textbf{s.t.} & c_1(s, l_1, a_1(s, l_1, t_2)) \geq 0 \\ & c_2(s, l_2, a_2(s, l_2, t_1)) \geq 0 \\ & t_1 = a_1(s, l_1, t_2) \\ & t_2 = a_2(s, l_2, t_1), \end{array} \tag{2}$$

where the disciplinary responses $a_1(s, l_1, t_2)$ and $a_2(s, l_2, t_1)$ are found by closing the disciplinary analysis constraints

$$\begin{array}{rcl} a_1 & = & A_1(s, l_1, t_2) \\ a_2 & = & A_2(s, l_2, t_1). \end{array}$$

The corresponding FIO formulation is

$$\begin{array}{cl} \underset{s,l_1,l_2}{\textbf{min}} & f(s, a_1(s, l_1, l_2), a_2(s, l_1, l_2)) \\ \textbf{s.t.} & c_1(s, l_1, a_1(s, l_1, l_2)) \geq 0 \\ & c_2(s, l_2, a_2(s, l_1, l_2)) \geq 0, \end{array} \tag{3}$$

where we compute $t_1(s, l_1, l_2)$ and $t_2(s, l_1, l_2)$ by solving the multidisciplinary analysis

$$\begin{array}{rclcrcl} a_1 & = & A_1(s, l_1, t_2) & \qquad & t_1 & = & T_1(a_1) \\ a_2 & = & A_2(s, l_2, t_1) & \qquad & t_2 & = & T_2(a_2). \end{array}$$

Thus, the DAO and FIO formulations (2) and (3) can be viewed as having been obtained from the SAND formulation (1) by closing some of the multidisciplinary consistency (analysis) equality constraints in the SAND formulation.

Other problem formulations eliminate local design variables by solving disciplinary optimization subproblems, as in Collaborative Optimization,[24,25,26,27] Optimization by Linear Decomposition,[28,29,30] and Optimization with Quasiseparable Subsystems.[31]

Sensitivities of the objective and constraints for DAO, FIO, and SAND can be shown to be connected via straightforward variable reduction operations.[21,22] These formulations have a special structure in the context of reduced basis optimization algorithms: computational components implemented for SAND can be reconfigured to yield the computational components needed for other formulations as long as feasibility with respect to certain analysis and consistency constraints is maintained (see Part2).

Manipulation of sensitivity components for use in other formulations, especially in multilevel optimization formulations, as easily as in DAO, FIO, and SAND in the context of reduced basis methods for optimization, is an open research area. However, computational components can certainly be re-used between problem formulations.

In order to ease problem formulation and solution by taking advantage of minimal computational components, we have to isolate these components. The smallest building bricks for our computational components will be the data exchanged among disciplines and optimization processes.

IV. Basic Tools of REMS

As mentioned previously, REMS relies on the abstraction of graph theory and on compiler-like language construction techniques. We model the flow of data among the disciplines or subsystems and the entities in the optimization process (objectives and constraints) as a directed graph, or digraph. The graph is then traversed to assemble information about the MDO problem in various contexts. The disciplinary practitioners need not be concerned with the mechanics of graph composition and operations. Instead, they provide data descriptions, at first autonomously, and then with a certain degree of interaction. REMS tools will read the descriptions and compose the graphs and the attendant tables. In order to enable the automatic conversion of raw data descriptions into graphs and tables, the descriptions have to conform to a set of formal rules, i.e., a grammar. In other words, we need a language to describe the data.

A directed graph consists of a set of nodes, a set of edges that connect the nodes, and two relations of incidence that define the direction of each edge. Any two nodes with an edge between them are distinguished as the head of the edge and the tail of the edge.

Our digraphs have two types of nodes: function nodes (denoted by squares in figures) and data nodes (denoted by circles). *Function nodes* represent operations. An operation may be a complex disciplinary analysis computation; an optimization of an entire subsystem; a conversion of some disciplinary outputs into optimization entities, such as constraints or objectives; or simple manipulation of input/output vectors, such as restriction or expansion, for use in another functional node. We also use function nodes to represent the constraints and objectives of optimization problems. *Data nodes* represent inputs and outputs of function nodes or functions. This assists us in expressing the dependence of data on data when assembling sensitivity information. Both function and data nodes may conceal a more complex structure or processes underneath. In that case, the nodes serve to aggregate appropriate operations.

A distinguishing feature of this representation is that both the functions and the data are regarded as nodes. In other publications, edges themselves often denote data. In our digraphs, edges are used to denote the direction of data flow. Further details of the graph representation can be found in Part 2 (the companion paper).

In order to assemble the graphs and manipulate information contained in them, the data description must be recorded according to a set of grammatical rules. The medium of data recording is the incidence matrix, a data structure that contains information about the node dependencies. The entry in the matrix is 1 if two nodes are connected by an edge and 0 if they are not. The incidence matrix works in conjunction with an additional data structure that gradually accumulates information about nodes (attributes) and their connections. The details of the language are beyond the scope of the two papers.

V. REMS Process

We describe the REMS process, illustrating the approach using a simple conceptual problem we have used elsewhere.[32, 5] Whereas in the earlier papers, the example started with the fully integrated formulation, here we approach the example at the outset from the perspective of autonomous disciplines, as we would for a realistic MDO problem.

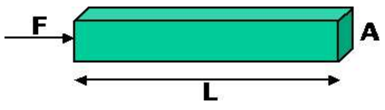

Figure 1. MDO problem with shared design variables.

Suppose two disciplines, stress $\mathcal{S}$ and weight $\mathcal{W}$, govern the behavior of a bar under a longitudinal load F. We want to describe these disciplines with a maximum degree of autonomy and then to reason about them in various contexts. We go through the process of building up the information about the problem step by step.

Step 1: autonomous disciplinary description.

Before "disciplines" communicate, all that discipline $\mathcal{S}$ knows is that it computes the stress on the bar, given load F and area A. That is, outputs and inputs of $\mathcal{S}$ are known, without reference to discipline $\mathcal{W}$. Similarly, all that discipline $\mathcal{W}$ knows is that it computes the weight of the bar, W, given density ρ, length L, and area A. These input–output relations are depicted in Figure 2.

At this stage, disciplinary experts $\mathcal{S}$ and $\mathcal{W}$, independently compose information about every node in the respective disciplines. The information about discipline $\mathcal{S}$ and discipline $\mathcal{W}$ may be assembled as in Tables 1 and 2. The actual descriptions will be simple lists of declarations.

Note that by allowing the disciplinary experts to describe their systems autonomously, without multidisciplinary considerations, we simplify their task to a great extent: the disciplinary graphs have a simple structure at this stage. This point will be especially pronounced in the more complicated example of the companion paper (Part 2), where the problem under consideration has four disciplines and many more inputs and outputs. There, the initial, strictly disciplinary graphs are visibly simpler than the multidisciplinary ones.

We also note that the disciplinary experts need not describe all their data at the outset, nor do they need

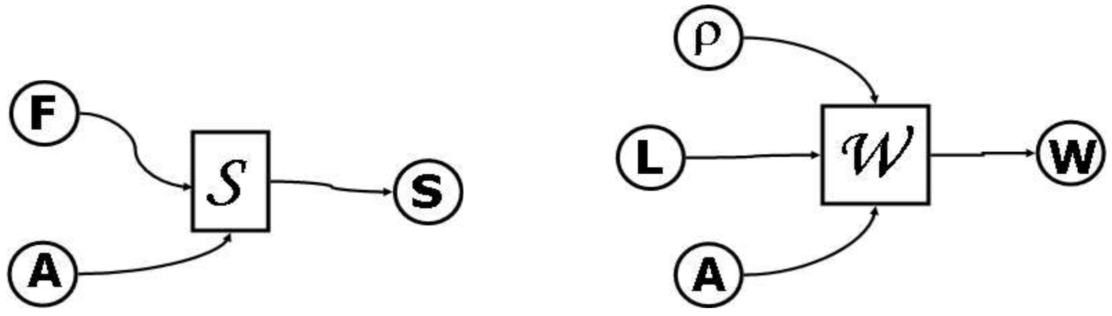

Figure 2. I/O of discipline $\mathcal{S}$ and discipline $\mathcal{W}$ in isolation.

Node	Description	O or I	Dimension	Continuity
A	cross-sectional area	input	1	continuous
F	longitudinal load	input	1	continuous
S	stress	output	1	continuous

Table 1. Output and input description for discipline $\mathcal{S}$.

Node	Description	O or I	Dimension	Continuity
ρ	density	input	1	continuous
A	cross-sectional area	input	1	continuous
L	length	input	1	continuous
W	weight	output	1	continuous

Table 2. Output and input description for discipline $\mathcal{W}$.

to include an exhaustive list of attributes. The intermediate representations of the problem are dynamic throughout the process, and information can be added or deleted, as needed.

At this stage REMS has I/O information about separate disciplines and it automatically assembles this information into intermediate representations – disciplinary graphs and tables. At this stage, also, automatic tools can compose the potential disciplinary sensitivity components.

Step 2: reconciliation of interdisciplinary coupling.

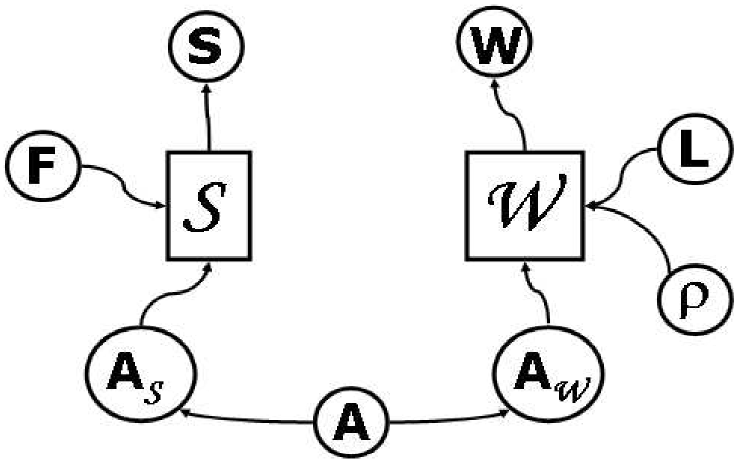

Figure 3. Input-output relations for coupled system without optimization.

At this stage, REMS assembles the disciplinary intermediate representations into a coupled multidisciplinary system. The process is akin to the linking process for computer code. The coupled system is depicted in Figure 3.

In this example, the disciplines are coupled through shared design variables, i.e., inputs. Instead of using a single input node for both functions, we explicitly specify that the data A feeds into disciplinary data nodes $A_{\mathcal{S}}$ and $A_{\mathcal{W}}$ to flag the opportunity for distributed computation. If the disciplines were also coupled through outputs, we would distinguished between the output node of one discipline and the input node of another, even if they were the same entity because such nodes also present an opportunity for distributed optimization formulations. This feature is illustrated in the example of Part 2.

At this stage, we can check for errors and also for intentions of the practitioners by examining the data requirements in the multidisciplinary context. For instance, if a function node expects a particular data input but REMS does not detect an output from an appropriate function node with an expected identifier (supplied by the attribute list), an error condition is flagged. Such error checking is not perfect, but in the presence of a large number of inputs and outputs, it is helpful.

In realistic applications, it is almost certain that disciplinary experts would have to communicate at this stage, at least by pairs, to reconcile input/output inconsistencies by updating the description lists. This stage can also reveal the need for introducing additional function nodes that serve for translating data among disciplines. Efforts in the development of interdisciplinary data dictionaries for specific applications[33] could assist in automating this stage further in the presence of a well-tuned data dictionary. And conversely, in the absence of a data dictionary, REMS could assist in compiling a data dictionary or a thesaurus.

Step 3: optimization problem.

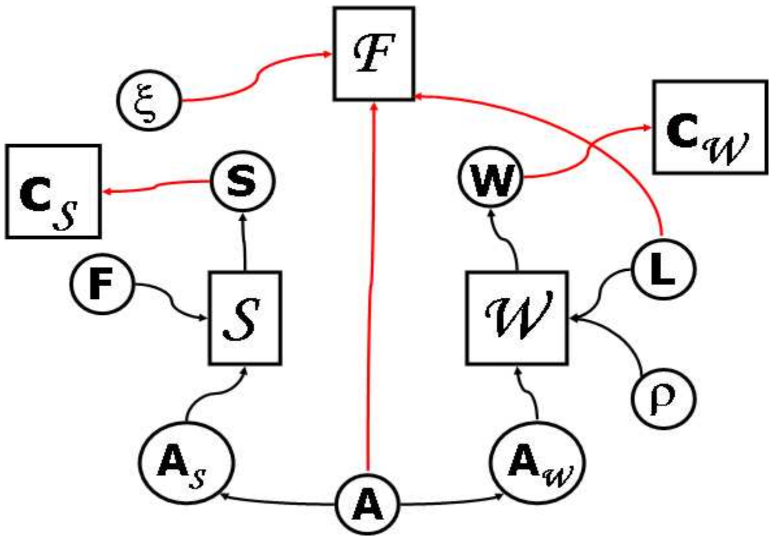

Figure 4. I/O for fully integrated optimization.

The graph in Figure 4 would correspond to the following optimization problem:

$$\begin{array}{lll} \underset{A}{\text{minimize}} & \mathcal{F} = \xi L A & \text{(total cost)} \\ \text{subject to} & S = F/A \leq \bar{S} & \text{(for some bound } \bar{S}) \\ & W = \rho L A \leq \bar{W} & \text{(for some bound } \bar{W}). \end{array}$$

Note, again, that the users would not be asked to compose the relatively detailed graph of Figure 4. Instead, they supply the much simpler disciplinary data descriptions in isolation. REMS then combines automated and interactive techniques to guide the representation to the data flow in an optimization formulation.

Once REMS establishes and reconciles the coupling, it can attempt to identify potential objectives and constraints by analyzing the status of the data nodes: all terminal nodes or leafs are potentially objectives and constraints. A user can then specify a problem formulation, and REMS can assemble the components into the necessary function and constraint information and the associated derivative. Multilevel optimization formulations present an interesting and difficult case. It involves the introduction of auxiliary data and function nodes. The principles are discussed id detail in Part 2.

VI. Concluding Remarks

REMS is a conceptual system for reasoning about complex systems in the decision making context, in general, and MDO, in particular. The fundamental goal is to ease system synthesis or integration and computational implementation by allowing the disciplinary practitioners to start the synthesis process by describing the inputs and outputs of their disciplines in as simple a language as possible, autonomously, without introducing multidisciplinary considerations early in the process. REMS then generates and analyzes a sequence of intermediate representations of the entered information, providing error checking and suggesting formulation alternatives to the user. Moreover, when the user makes changes in the problem components, REMS can propagate these changes throughout the problem representation. We have outlined the general methodology in the two companion papers. The details of the abstract language and the analysis and manipulation of representations are the focus of our current research.

References

[1]Alexandrov, N. M. and Lewis, R. M., "Reconfigurability in MDO Problem Synthesis, Part 1 and Part 2," August 2004, Papers AIAA-2004-4307 and AIAA-2004-4308.

[2]Thareja, R. and Haftka, R. T., "Numerical difficulties associated with using equality constraints to achieve multi-level decomposition in structural optimization," AIAA Paper 86-0854, 1986.

[3]Kodiyalam, S., "Evaluation of Methods for Multidisciplinary Design Optimization, Phase I," Tech. Rep. NASA/CR-1998-208716, NASA Langley Research Center, September 1998.

[4]Alexandrov, N. M. and Kodiyalam, S., "Initial Results of an MDO Method Evaluation Study," 1998, AIAA Paper 98-4884.

[5]Alexandrov, N. M. and Lewis, R. M., "Analytical and Computational Aspects of Collaborative Optimization for Multidisciplinary Design," *AIAA Journal*, Vol. 40, No. 2, February 2002, pp. 301–309.

[6]Alexandrov, N. M. and Lewis, R. M., "Analytical and computational properties of distributed approaches to MDO," AIAA Paper 2000-4718, September 2000.

[7]Alexandrov, N. M. and Lewis, R. M., "Algorithmic perspectives on problem formulations in MDO," AIAA Paper 2000-4719, 2000.

[8]Frank, P. D. and Shubin, G. R., "A comparison of optimization-based approaches for a model computational aerodynamics design problem," *Journal of Computational Physics*, Vol. 98, No. 1, January 1992, pp. 74–89.

[9]Cramer, E., Dennis, Jr., J. E., Frank, P., Lewis, R., and Shubin, G., "Problem formulation for multidisciplinary design optimization," *SIAM Journal on Optimization*, Vol. 4, No. 4, November 1994, pp. 754–776.

[10]Benson, S., McInnes, L. C., Moré, J., and Sarich, J., "TAO Users Manual," Technical Report ANL/MCS-TM242-Version1.7, August 2004, Mathematics and Computer Sciences Division, Argonne National Laboratory.

[11]Fourer, R., Gay, D. M., and Kernigan, B. W., *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press, 1993.

[12]Harary, F., *Graph Theory*, Addison–Wesley, 1994.

[13]Cooper, K. D. and Torczon, L., *Engineering a Compiler*, Morgan Kaufmann Publishers, 2004.

[14]Griewank, A., *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers in Applied Mathematics, SIAM, Philadelphia, 2000.

[15]Wagner, T. C., "A General Decomposition Methodology for Optimal System Design," Dissertation, Department of Mechanical Engineering, University of Michigan, 1993.

[16]Etman, L. F. P., Kokkolaras, M., Papalambros, P. Y., Hofkamp, A. T., and Rooda, J. F., "Coordination specification of the analytical target cascading process using the χ language," *9th AIAA/ISSMO Symposium of Multidisciplinary Analysis and Optimization, Atlanta, GA*, 2002, AIAA-2002-5637.

[17]Etman, L. F. P., Hofkamp, A. T., Rooda, J. F., Kokkolaras, M., and Papalambros, P. Y., "Coordination specification for distributed optimal system design using the χ language," *9th AIAA/ISSMO Symposium of Multidisciplinary Analysis and Optimization, Atlanta, GA*, 2002, AIAA-2002-5410.

[18]Rogers, J. L., "DeMAID/GA An Enhanced Design Manager's Aid for Intelligent Decomposition," *Presented at the Sixth*

AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, Washington, 1996, AIAA paper 96–4157.

[19]Phoenix Integration, Inc., “Improving the Engineering Process with Software Integration,” 2002, a white paper.

[20]Eldred, M. S., Swiler, L. P., Gay, D. M., Brown, S. L., Giunta, A. A., Wojtkiewicz, S. F., Hart, W. E., and Watson, J., “DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis, Version 3.2 Reference Manual,” Sandia National Laboratories, http://endo.sandia.gov/DAKOTA/software.html, 2004.

[21]Lewis, R. M., “Practical Aspects of Variable Reduction Formulations and Reduced Basis Algorithms in Multidisciplinary Design Optimization,” *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. Alexandrov and M. Y. Hussaini, SIAM, Philadelphia, 1997.

[22]Alexandrov, N. M. and Lewis, R. M., “Dynamically Reconfigurable Approach to Multidisciplinary Problems,” June 2003, AIAA Paper 2003-3431.

[23]Haftka, R. T. and Gürdal, Z., *Elements of Structural Optimization*, Kluwer Academic Publishers, Boston, 1993, 3rd edn.

[24]Braun, R., *Collaborative Optimization: An architecture for large-scale distributed design*, Ph.D. thesis, Stanford University, May 1996, Department of Aeronautics and Astronautics.

[25]Braun, R. D., Moore, A. A., and Kroo, I. M., “Collaborative approach to launch vehicle design,” *Journal of Spacecraft and Rockets*, Vol. 34, July 1997, pp. 478–486.

[26]Braun, R. D. and Kroo, I. M., “Development and application of the collaborative optimization architecture in a multidisciplinary design environment,” *Multidisciplinary Design Optimization: State of the Art*, edited by N. M. Alexandrov and M. Y. Hussaini, SIAM, 1997, pp. 98–116.

[27]Sobieski, I. and Kroo, I., “Aircraft design using collaborative optimization,” AIAA paper 96-0715, presented at the 34th AIAA Aerospace Sciences Meeting, Reno, Nevada, Jan. 15-18, 1996, 1996.

[28]Sobieszczanski-Sobieski, J., “A linear decomposition method for large optimization problems—blueprint for development,” Tech. Rep. TM 83248, NASA, 1982.

[29]Barthelemy, J.-F. M., “Development of a Multilevel Optimization Approach to the Design of Modern Engineering Systems,” NASA Contractor Report 172184, August 1983, Virginia Polytechnic Institute and State University, Blacksburg, VA.

[30]Sobieszczanski-Sobieski, J., James, B. B., and Riley, M. F., “Structural Sizing by Generalized, Multilevel Optimization,” *AIAA Journal*, Vol. 25, No. 1, January 1987, pp. 139–145.

[31]Haftka, R. T. and Watson, L. T., “Multidisciplinary Design Optimization with Quasiseparable Subsystems,” Optimization and Engineering, December 2004, to appear.

[32]Alexandrov, N. M. and Lewis, R. M., “Analytical and Computational Aspects of Collaborative Optimization,” National Aeronautics and Space Administration, NASA/TM–210104–2000, April 2000.

[33]Vander Kam, J., “The launch vehicle language (LVL) data model for evaluating reusable launch vehicle concepts,” January 2003, 41st Aerospace Sciences Meeting & Exhibit, Reno, Nevada, AIAA-2003-1330.