Development of a Distributed Design Toolkit for Analyzing Process Architectures

E. D. Devendorf[1], P. Cormier[2], and K. E. Lewis[3]
University at Buffalo - SUNY, Buffalo, NY, 14260

The distribution of a complex system design problem into smaller, coupled subsystem problems creates a number of research and implementation challenges, including determining the most appropriate process architecture. Here the term *process architecture* refers to the order in which design subsystems solve their individual optimization problem. While methods have been proposed to model specific distributed process architectures, analyzing the impact of a wide range of process architectures on solution quality, convergence, and stability still remains a difficult task. To address this, we introduce a toolkit that enables the modeling and analysis of distributed design process architectures using basic sequential, parallel, and hybrid elements. Features of the toolkit are described and its use in studying solution characteristics including quality, convergence, and stability is discussed. The toolkit allows for users to create and analyze possible processes architectures using graphical interfaces.

I. Introduction

In modern engineering, successfully meeting design requirements often requires the involvement of a variety of experts. These experts may originate from radically different disciplines or be specialized in different aspects of the same discipline. In both cases, it is typical for an expert to only fully understand a very specific aspect of the aggregate design problem. To facilitate communication and cooperation between these experts a variety of design frameworks have been created that fall under the auspices of Multidisciplinary Design Optimization (MDO) [1-4].

The factors contributing to which MDO framework is most appropriate for a particular system depends on the complexity of the system being analyzed, the availability of experts and the resources available to the designers. Perhaps the most influential of these three factors is the increasing complexity and size of modern designs. Changes in the automobile industry highlight the increasing size of design systems. Ford's original Model T was composed of roughly 700 different parts. In contrast modern vehicles have more parts in their radio alone [5]. An even more extreme case is the Boeing 777, which has over 3,000,000 parts provided by over 900 different suppliers [6].

In addition to the increase in system size, modern designs are also becoming increasingly sophisticated. At its inception the Space Shuttle was one of the most advanced systems ever conceived and it utilized unique drive-by-wire technology [7]. This same technology is now common in many ordinary automobiles [8]. Incorporating modern technological innovations into new products requires expertise that is often not traditionally associated with those products. Growing or maintaining an engineering staff that contains all that knowledge is also a major challenge. One of the more common solutions to this challenge is to outsource to suppliers a portion of the design responsibility traditionally shouldered by a single organization [9].

Although outsourcing a portion of the design responsibility addresses some challenges associated with product design, it also introduces new obstacles in coordinating between many different design teams. With advances in cyberinfrastructure [10, 11] and information technology these design teams are often separated both geographically and temporally. As cyberinfrastructure and information technology continue to evolve, so will both the separation and the size of design networks.

One of the best modern examples of the consequences of failing to appropriately manage a complex MDO design network is captured in the experiences of the aerospace companies Boeing and Airbus. Boeing has faced major challenges during the design and assembly of their new 787 Dreamliner aircraft [12]. In an effort to mitigate risk and shed development costs Boeing shifted a large portion of the overall system design to its suppliers.

[1] Research Assistant, Mechanical and Aerospace Engineering Department, University at Buffalo
[2] Research Assistant, Mechanical and Aerospace Engineering Department, University at Buffalo
[3] Professor, Mechanical and Aerospace Engineering Department, Associate Fellow AIAA, kelewis@buffalo.edu

However, challenges coordinating between these suppliers has rippled across Boeing's entire supply chain resulting in large financial losses, as well as, continued product delivery delays.

One mitigating factor for Boeing has been the similar troubles faced by their major competitor Airbus. Airbus was formed as a conglomerate of many smaller companies within the European Union and is naturally separated by geography, culture and language. Difficulties coordinating between its different divisions and its suppliers have caused delays in the Airbus's new plane, the A-380. Their delay cost Airbus both monetarily with its clients and in the opportunity to undermine the market share of one of its competitors [13].

In order for Boeing and Airbus to remain competitive into the twenty-first century, neither organization can afford to repeat the mistakes made during the development of the 787 Dreamliner and A-380 respectively. One of the contributors to the setbacks faced by Boeing and Airbus was in managing their design process. With the natural increase in system complexity and size, other organizations will face similar obstacles to those of Boeing and Airbus.

One of these obstacles is developing an understanding of the role of time to convergence in MDO problems. In the same way time studies are critical to manufacturing processes, awareness of the time required to design a complex system is critical. This paper presents a toolkit that provides support to analyze dynamic system behavior, an important step towards developing optimal system architectures. This toolkit is designed to operate within existing distributed design frameworks. It utilizes recent advances in the understanding of distributed design to predict convergence rates based on the system's architecture properties [14, 15].

The approach provides a visual tool for increased design process control through an understanding of how the distributed design problem structure influences the time required to reach a solution. This provides support for designers to set realistic design deadlines, establish timelines for product development, and control costs during design. The next section provides additional background on MDO frameworks and provides context for the choice of a distributed design MDO framework.

II. Multidisciplinary Design Optimization Frameworks

There are two primary classifications for MDO frameworks based on the process used to arrive at a final design [16]. From a structural perspective the simplest classification is an all-in-one approach [4]. In an all-in-one approach designers from different disciplines cooperate with one another to formulate a single aggregate objective function with associated constraints to reach a globally optimal solution. An all-in-one approach has significant advantages. There is only a single aggregate problem to address, all designers are working towards a single unified goal, and information about the entire system is available to any designer.

The advantages of an all-in-one approach make it attractive; however, this approach is often challenging to apply to large, complex design systems. Further, there are often situations where an all-in-one approach is not feasible due to system constraints (*e.g.,* suppliers will not share design information). Due to the necessity of system decomposition, designers have a wide range of design tools to assist in this task. The process employed by these tools can typically be decomposed into two fundamental steps: (1) identify necessary subsystems and (2) create a framework for communication between these subsystems. The first step in this process is not the topic of this paper, but remains an important and challenging research topic [17]. Some of the techniques used to identify the necessary subsystems were surveyed by Sobieski and include: object decomposition, aspect decomposition, sequential decomposition and model decomposition [18].

Once the required subsystems have been identified in Step 1, a design "framework" is created. In this paper "framework" refers to the specific MDO approach used to solve the problem. In each framework there is a set of rules that govern the subsystem objective functions, communication protocol, assigning control of design variables and the other aspects necessary to successfully complete the design process. There are a wide range of MDO frameworks that make guarantees with respect to system convergence and optimality. Some of these frameworks include Concurrent Subspace Optimization [3], Analytical Target Cascading [19], Bilevel Integrated System Synthesis [20] and Collaborative Optimization [21].

Each of these frameworks has its own advantages. For example, in Analytical Target Cascading the system is guaranteed to converge to a globally optimal solution [22]. In spite of the advantages to adopting a framework, there are still many cases when one is not chosen. Without a framework a strong system level presence can be lacking, and designers may complete design tasks more often based on their own needs. Further, communication is based upon the information required by each designer to complete these tasks.

There are several reasons why a formal framework may not be applied in an MDO problem. Applying an MDO framework requires: a large degree of sophistication and expertise on the part of the design managers and the design subsystems; all the subsystems must agree in the first place to adopt a framework; and some design problems do not

naturally lend themselves to formal frameworks. When a formal framework is not chosen or when the framework does not specifically proscribe subsystem interactions, MDO problems often default to distributed design problems. In general the organization, design subsystem requirements and communication protocols in distributed design are less sophisticated than other MDO frameworks. However, this does not make analyzing distributed design systems an easy task. The assumptions and mechanics governing distributed design systems are discussed in greater depth in Section III.

III. Distributed Design

A. Problem Structure

Distributed design problems are a specific type of MDO problem and distributed design mechanics are an important topic in design research [23, 24]. By their nature, distributed design problems are non-hierarchical and each design subsystem individually seeks to minimize their own objectives. In some cases distributed design problems are cooperative and when this occurs they are similar to the all-in-one approach to MDO problems. However, when they are non-cooperative they have more in common with decomposition frameworks. Although distributed design has more in common with decomposition frameworks when it is non-cooperative in nature, it is not necessarily true that all decomposition approaches are non-cooperative. For example, Concurrent Subspace Optimization and Collaborative Optimization have many of the characteristics of a cooperative framework [25].

Non-cooperative distributed design problems are common in engineering practice. In order for an MDO problem to qualify as a distributed design problem it must have the following properties [26]:

1. Designers have knowledge of only their own objective function,
2. Designers act unilaterally to minimize their own objective function,
3. Designers have complete controls over specific local design variables,
4. Designers communicate by sharing the current value of their local design variables.

There are many cases when distributed design problems emerge as sub-problems from applying one of the other MDO frameworks discussed in Section II. As long as the sub-problem is iterative and has the properties listed above, its behavior will be similar to an ordinary distributed design problem. For example, one approach to system decomposition uses design structure matrices (DSM's) to represent a system [27]. The goal of this approach is to reduce the number of feedback loops between sub-systems, but it is often the case that they cannot all be completely eliminated. Each of these feedback loops can be considered as a discrete distributed design problem and modeled using the applicable theory. This design approach has also been automated by Bloebaum and Rogers with a software tool called the Design Manager's Aid for Intelligent Decomposition (DeMAID) [28]. In Section IV, the differences between the developed toolkit and DeMAID are discussed. In the next section the game theoretic background of equilibrium and distributed deign stability are discussed.

B. Equilibrium Stability

A significant portion of the work understanding and modeling the mechanics of distributed design has been accomplished through the application of game theoretic models to a design process. This approach was first introduced by Vincent [29] who examined equilibrium conditions in distributed design. In Vincent's model the assumptions of non-cooperation were applied to model distributed design systems. Using these assumptions Vincent demonstrated that the designers can be modeled as players and after iteration converge to the Nash Solution, also known as the non-cooperative equilibrium [30]. Vincent's approach was revised by Chanron in [31] to model distributed design systems using Systems Theory [32] and a state space representation. In [33], it was demonstrated that design systems can have multiple Nash Equilibriums, which can each be individually stable, unstable or saddle points [14]. In [34], the approach was extended using nonlinear system theory and Lyapunov functions to identify convergent regions of the design space. For convergent Nash solutions these regions are called the equilibrium's domain of attraction and the approach was later refined to provide a better approximation in [35].

Fundamental to understanding convergence in distributed design is the idea of Rational Reaction Sets (RRS). An RRS is the partial derivative of a designer's objective function with respect to the design variables it controls. The intersection of the designer's RRS's is the system's Nash Equilibrium. A simulated example of an iterative design process is shown below in Figure 1. In this simulation designer 1 controls design variable x and designer 2 controls design variable y. Since the designers in this case are applying an iterative process, each intersection marked by an *'x'* in Fig. 1 represents a single designer iteration. The starting point for the simulated example is $(x,y)=(1,4)$ and designer 1 and designer 2's objective functions are shown in Eqns. (1) and (2) respectively.

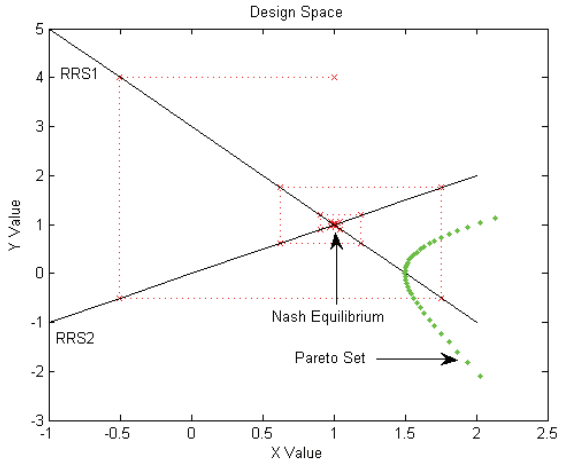


Figure 1: Sample Sequential Convergence Plot

In Fig. 1 the designers converged after 26 iterations to the Nash Equilibrium, market by the intersection of both designers' RRS's. The points in the bottom right hand corner of Figure 1 are the Pareto Set and while these points are important in the investigation of optimality for Nash Equilibriums, it does not influence the system's convergence behavior.

$$F_1 = x^2 - 3x + xy \quad (1)$$
$$F_2 = 0.5y^2 - xy \quad (2)$$

Research into distributed design has primarily emphasized investigating the stability characteristics of the overall system. However, in [15] a formulation was presented to examine the transient response for two designer systems like the one shown in Eqns. (1) and (2). This work found that the speed of the transient response is related to the solution process architecture chosen. Section C discusses distributed design solution process architectures in greater detail, and it is shown how changes in solution process architecture can change the convergence time of the two designer problem in Eqns. (1) and (2).

C. Design Architecture

In this paper whenever the term "architecture" is used it refers to solution process ordering of the system. It does not refer to product architecture, which is an independent and significant area of design research [36]. This process structure includes both sequential and simultaneous elements. An example of a purely sequential and purely simultaneous architecture is shown in Fig. 2 along with a hybrid architecture that incorporates aspects of both.

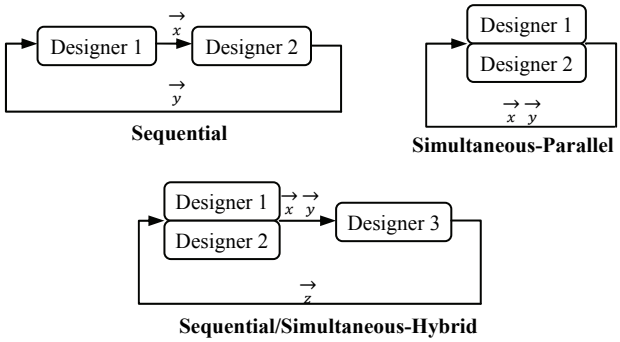


Figure 2: Design Architectures

Past research into distributed design convergence has not focused on the solution process architecture chosen for the system. In Chanron's formulation it was assumed all designers iterated sequentially, and it was assumed that the order in which the process is executed does not change the equilibrium stability. For simultaneous or parallel systems, the same result was demonstrated by Smith and Eppinger [37]. Although it has not been shown that

stability changes with solution process architecture, it can be shown that the system's transient response changes based on these choices.

For the design system summarized in Eqns. (1) and (2) and shown in Figure 1 the design subsystems were sequentially ordered and converged to within 2% their settled value after 26 iterations. When the same two designer system is simulated again with subsystems 1 and 2 in parallel and with the same convergence criteria, the path taken to the Nash Equilibrium is much different. In contrast to the sequential architecture, the parallel architecture converges in 22 iterations. The step by step process for a parallel architecture is shown in Fig. 3.

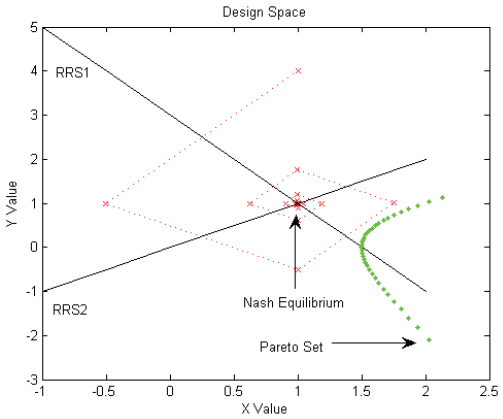


Figure 3: Sample Parallel Convergence Plot

Therefore, even for a two designer system there is a significant change, 18%, in the number of iterations required for the sequential architecture to converge when compared to the parallel architecture. It is important to emphasize, however, that in the sequential architecture each iteration includes a single minimization while in the parallel architecture two minimizations are required, one for each designer. The resulting increase in convergence rate, therefore, came at the cost of 70% more minimizations. In Section D the current state of research addressing the transient response of distributed design systems is discussed.

D. Transient Response

While the primary focus of distributed design research has been on equilibrium stability, there has been significant research investigating transient response as well. In [28, 39] Rogers made use of the DeMAID software package and the global sensitivity equations [38] to minimize the number of individual designer iterations required by selectively skipping some systems that were designated as having a low sensitivity to the current design task. Also, in [40], the relationship between removing weak couplings for multiple iterations and the efficiency and quality of the resulting solution is studied. These approaches reduced the computational time required to reach a solution, but did so by fundamentally altering the subsystem communications. In distributed design problems, this level of direct control over subsystems is not always available. Further, this approach is developed only for sequential design orderings and there are cases when all the criteria outlined in the approach are fulfilled but multiple design orderings with different convergence times are still possible.

Another contribution to examining convergence rate applied a work transformation matrix, analogous to the global sensitivity equations, to link design tasks in [37]. This approach was designed for strictly parallel systems and relative designer sensitivities were determined using an eigenvalue analysis. While this process explores the basic mechanisms governing convergence, it does not suggest optimal design orderings or predict convergence time. In [15] a direct approach was taken to determining convergence rates for an unconstrained two designers, two design variables problem. In this analysis a closed form expression was derived to determine the number of iterations required to achieve convergence. This formulation was applicable to both sequential and parallel solution process architectures, but is limited in scope to small design problems. In this paper an approach is used that incorporates control theory to examine the transient response of design systems. The toolkit where this approach is implemented is presented in Section IV.

IV. Distributed Design Toolkit

The Distributed Design Toolkit (DDT) provides 3-Dimensional (3-D) visual support to design managers analyzing distributed systems. Although there has been significant fundamental research in the field of distributed

design, this work focuses more on the application of this research to develop tools capable of supporting decision making in distributed design processes. The DDT increases the accessibility of this research, allowing design managers to easily test different solution process architectures. For example, during a disaster, emergency response services are often grouped into autonomous disciplinary teams. Organizing these decision making teams properly can decrease overall transient response time.

The DDT is unique in that its focus is on distributed design systems, but it is important to differentiate it from existing software packages that have been developed to assist design managers. The approach taken by two of these packages, DeMAID and Visual Dependency Structure Matrix (VDSM), address issues similar to the DDT. Both DeMAID and VDSM utilize DSM's to organize design systems. In DeMAID DSM's are automatically generated based on a set of input designers [27]. The global sensitivity equations were added in [28] along with a genetic algorithm to reduce overall design time.

VDSM distinguishes itself from DeMAID by using three-dimensional DSM's. To reduce computational cycle time, VDSM uses design couplings with sensitivity analysis in [41] to identify system reduction and sequencing techniques. VDSM is written in an XML format and is therefore platform independent and can be used collaboratively through the internet. The DDT is different from DeMAID and VDSM in four ways. The first is that the DDT has been specifically created for use with distributed design systems. Secondly, the DDT does not use DSM's to organize designers, and instead organizes them in a 3-D space where their position in space dictates the solution process architecture. The third difference is that the DDT enables designers to be explicitly arranged in parallel with one another. The last characteristic is that the DDT uses techniques specific to distributed design that enable the determination of equilibrium stability and transient response rates without simulating the design system [31, 32].

Similar to DeMAID and VDSM, the DDT is flexible in its use and can be used on any Windows XP or newer system. The DDT is written in C++ and the graphics processing is performed by OpenGL. It requires no proprietary computer hardware or software, but it does leverage the MatLab© Compiler and Matrix TCL Lite packages. Both these software packages are available free from their respective distributors and the MatLab© Compiler can be bundled with distributions of the DDT. The DDT merges fundamental mathematical equations and approaches for evaluating distributed design systems with a point and click interface to enable users to create, manipulate, and evaluate design architectures. Once a distributed design system has been created, it can be saved for later use or analysis, and loaded at a later date when needed.

In the following sections a walkthrough of the DDT is provided. In Section A the graphical interface is reviewed and the designer properties explained. Section B details the specifics of architecture creation and design. Finally, Section C demonstrates the analysis tools available with the current deployment of the DDT.

A. Graphical Interface

The DDT is initialized with a single subsystem, represented by a blue cube located at the origin, and an associated grid that stretches across the *x-y* plane, shown in Figure 4. Users can either immediately begin creating a new system or load existing design architectures through the *"File*" menu using the *"Open"* command. To create a new process architecture, the first step is to modify the initial design subsystem. Subsystems have six distinguishing characteristics: name, color, size, location, objective function, and controlled design variables. The first four of these characteristics exist to visually distinguish between subsystems, while the latter two determine the subsystems' mathematical properties. The user has direct control over the visual and mathematical properties of the designers. The definitions and limitations of these properties are summarized in Table 1.

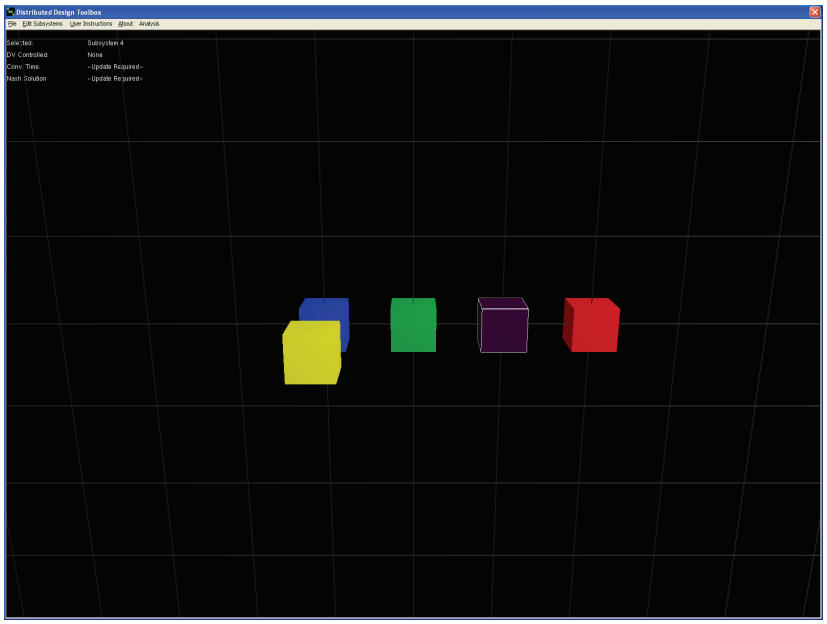


Figure 4: Program Overview Example

Table 1: Subsystem Properties

Property	Description
Name	Provides each subsystem with an identity. Names can be 64 keyboard characters long and the default name is Designer X.
Color	Distinguishes each subsystem by assigning a color. Colors are defined by specifying a value between zero and one for each color component. These components include red, green and blue. The default color is blue.
Size	Size is a default value which controls the relative proportions of each subsystem. Future versions of the DDT will use this variable to emphasize important subsystems
Location	Each subsystem can be moved to any point in the *x-y* coordinate system. Subsystems are automatically snapped to the nearest gridline intersection. Stacking subsystems in the *z*-direction dictates parallel solution process architectures.
Objective Function	An unconstrained objective function is assigned to each designer. This objective function currently includes quadratic and linear terms as shown in Figure 7.
Controlled Design Variable(s)	The variables controlled by a designer are stored as a vector. These variables are defined as *DV1* through *DV20*, and specified by checking the appropriate box in Figure 7.

To create a new design subsystem or to edit an existing subsystem the user can select the *'Edit Subsystems'* menu from the menu bar. The *'File'* menu opens a drop down list of commands to create a new system, open an existing system or save the current system. The two menu bar options *"User Instructions"* and *"About*" launch windows containing user instructions and information about the current version of the program respectively. The last menu option "*Analyze"* prompts the user for analysis options and allows the analysis to be launched. The menu controls are summarized in Table 2.

Table 2: Menu Controls

Menu Option	Description of Functionality
File	New: *Creates a new document* Open: *Opens a previously saved document* Save: *Saves current distributed design system* Save As: *Allows the user to save the current distributed design system as a new system* Recent Files: *Displays recent files for quick access* Exit: *Exits the program; unsaved data will be lost*
Edit Subsystems	Launches the *Add/Edit/Delete Subsystems* dialog window (Figure 5), which allows the user to add, edit, or remove subsystems. Editing allows the user to modify the subsystem's objective function and design variable control, as well as display information such as color and position. New subsystems can also be created (Figure 6).
User Instructions	Launches the *User Instructions* dialog window (Figure 9), which contains directions and controls for the program.
About	Launches the *About* dialog window with information about the creation of the program.
Analyze	Launches the *Distributed Design Analysis* dialog window to select analysis options and initiate the analysis of the distributed design system.

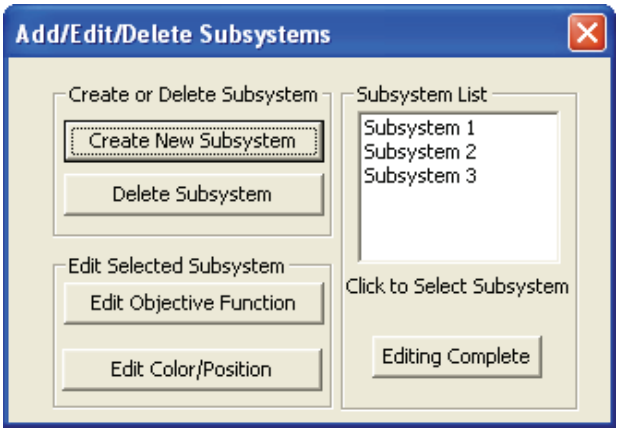


Figure 5: DDT Add/Edit/Delete Subsystem Dialog

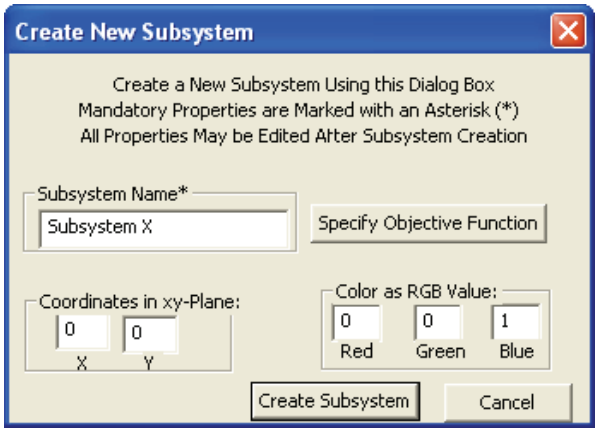


Figure 6: Create New Subsystem Dialog

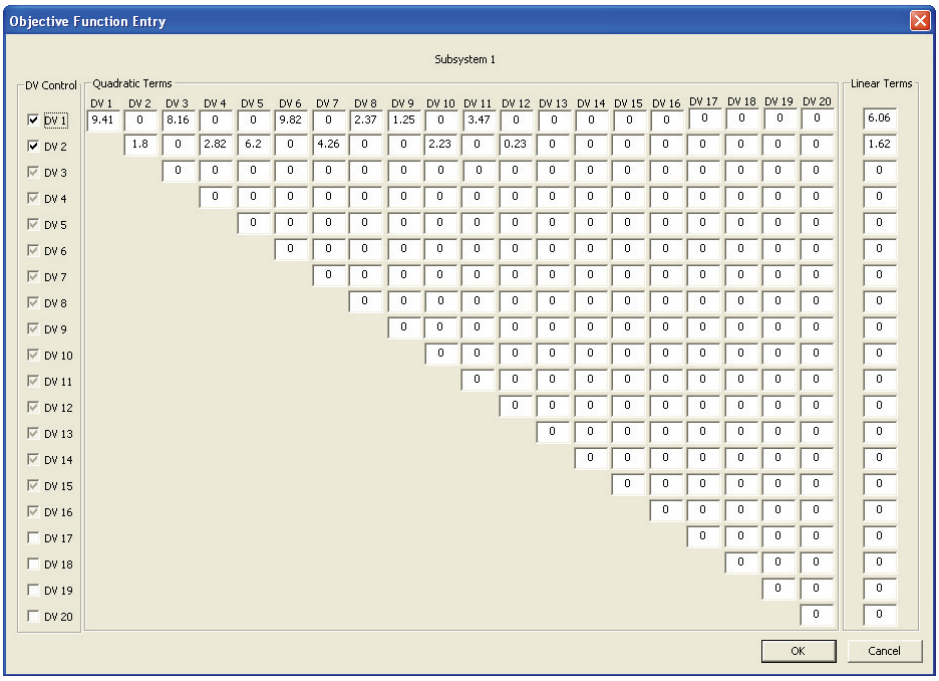


Figure 7: Objective Function Entry Example

To add, edit or delete a subsystem the user selects the men option *"Edit Subsystems,"* which launches a dialog window with the form shown in Figure 5. To create a new subsystem the user clicks on the *'Create New Subsystem'* button, which launches the window shown in Figure 6. Here the user can set the name, position, color, and launch a window for inputting the subsystem objective function by selecting the *"Specify Objective Function"* button.

The Objective Function Entry window (Figure 7) allows the users to input the coefficients for the second order and linear terms for the objective function and specify which design variables the subsystem controls. Design variables already assigned to a different subsystem are grayed out, as only one subsystem can control them for distributed design systems. The objective function shown in Figure 7 for Subsystem 1 is equivalent to the following:

$$f_1(x) = 9.41x_1^2 + 1.80x_2^2 + 8.16x_1x_3 + 2.82x_2x_4 + 6.20x_2x_5 + 9.82x_1x_6 + 4.26x_2x_7 + 2.37x_1x_8 + 1.25x_1x_9 + 2.23x_2x_{10} + 3.47x_1x_{11} + 0.23x_2x_{12} + 6.06x_1 + 1.62x_2$$

Once the objective function is set, the user can click on the *"Create Subsystem"* button, and the new subsystem is created. For subsystems that already exist, the user can select one from the subsystem list, and then select *"Edit Objective Function"* or *"Edit Color/Position"* to modify the subsystem. The *"Edit Objective Function"* button launches same window as is used for the initial objective function input. The *"Edit Color/Position"* button brings up the window shown in Figure 8.

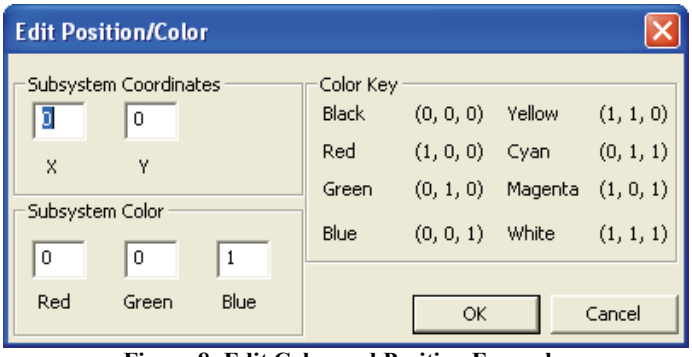


Figure 8: Edit Color and Position Example

The number of allowable unique distributed design subsystems and the number of total design variables is twenty. However, a minimum of two user-defined subsystems is required to execute any analysis. Providing effective navigation controls for potentially large architectures is crucial for easy subsystem manipulation and

viewing point adjustment. Therefore, functionality is based on simple mouse and keyboard commands and these are highlighted in the User Instructions window, Figure 9. When a designer is selected, they can be dragged to a new location on the grid by left clicking. Additionally, the mouse can be used to rotate the scene about the *x*, *y* and *z* axis; the mousewheel allows the user to zoom in or out. These controls were designed to be similar to common 3-D modeling programs. In the next section the specification of a process architecture is discussed.

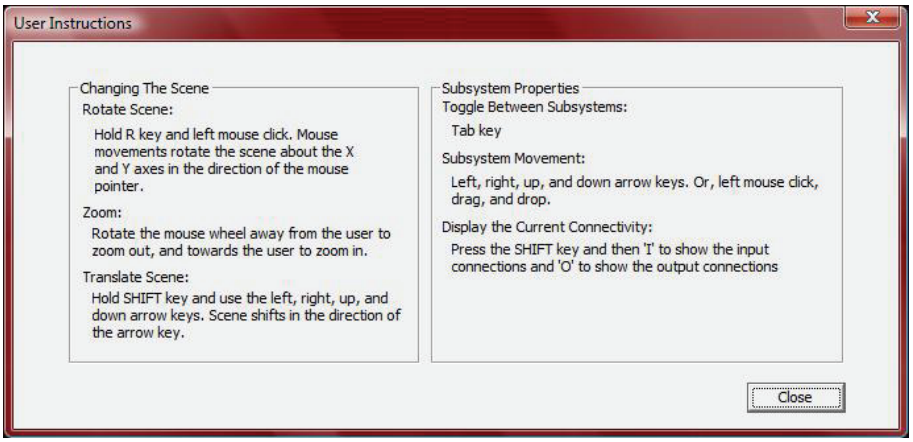
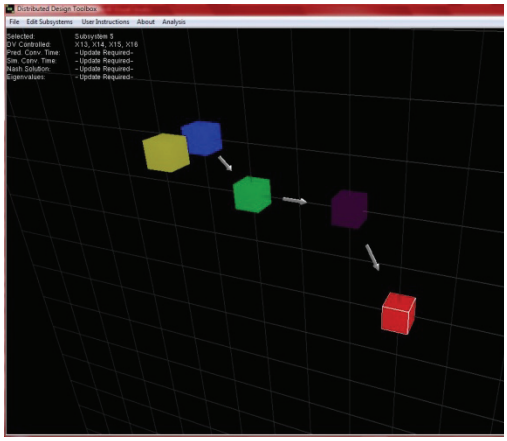


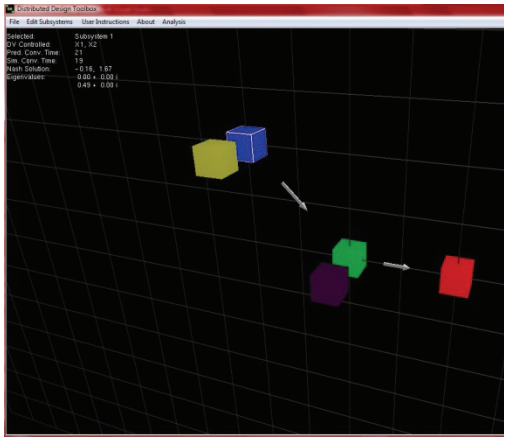
Figure 9: User Instructions Pane

B. Architecture Specification

When a process architecture contains two or more subsystems, the user can specify the desired architecture. Sequential connectivity is governed by the relative *X* and *Y* coordinates of the subsystems. The order of the subsystems is determined first by the left to right position in the *X* direction, with subsystems to the left going first. If two subsystems have the same *X*-position, the order is determined from top to bottom in the *Y* direction. If two designers have the same *X* and *Y* position, the designers are in parallel, and will be placed in different *Z* locations. To view the current subsystem ordering, the user can display the ordering by using the command *SHIFT+A*. This will show the current connectivity with white arrows, Figure 10.



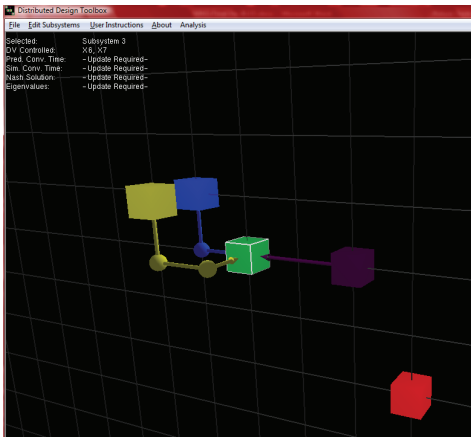
Two Subsystems arranged in parallel with the remaining subsystems arranged sequentially



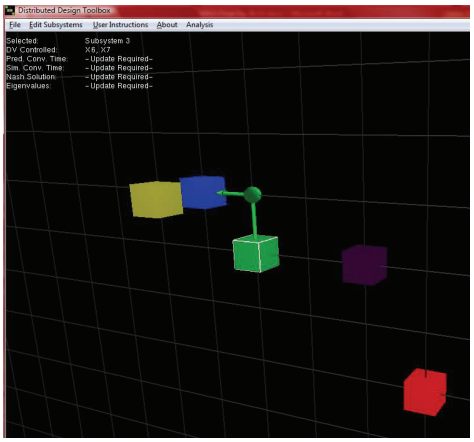
Two pairs of subsystems arranged in parallel with the remaining subsystem arranged sequentially

Figure 10: Designing the Overall System Architecture

The white arrows in Figure 10 are a visual aid to verify that the desired architecture has been specified. When specifying architectures, however, additional information about the relationships between the subsystems can also be shown. The commands shift-I and shift-O create connectors that show which other subsystems have inputs into the selected subsystem or the other subsystems affects by the selected subsystems respectively. A subsystem receives input from another if a design variable controlled by the other subsystem has a non-zero coefficient in its Rational Reaction Set. The input and output parameters for the architecture on the left of Figure 10 is shown for subsystem 3 in Figure 11.



Subsystems with Inputs into Subsystem 3



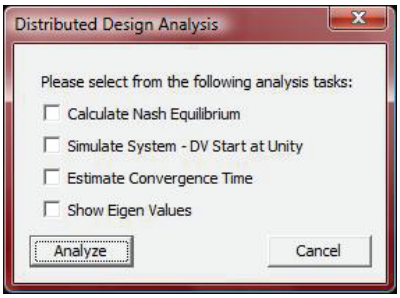
Subsystems with outputs from Subsystem 3

Figure 11: Input/Output Connectors

The input connections on the left side of Figure 11 are colored according to the subsystem from which they originate. From this representation it is clear that subsystem 3 takes inputs from subsystems 1, 2 and 4 (*blue*, *yellow*, *purple*) while outputting design variable values that effect subsystem 1 (*blue*). Using only the input/output relationships between the subsystems can help design managers to create subsystems with the desired transient characteristics. An approach using these relationships is presented in [42]. In the following section the architecture specified on the left of Figure 11 is examined analytically using the tools provided by the DDT.

C. Analyzing Distributed Design Systems

Once the process architecture has been specified for a populated set of design subsystems the DDT provides analysis tools to determine the overall system behavior. These tools can be access through the *Analyze* menu and are controlled through the dialog shown on the right of Figure 12.



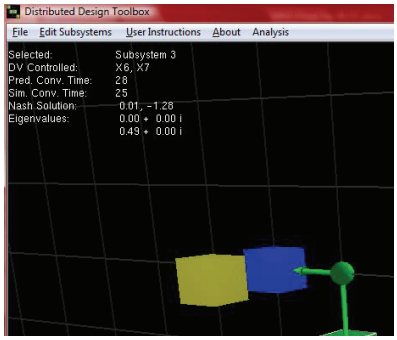


Figure 12: Distributed Design Analysis Dialog

The dialog on the left of Figure 12 provides four analysis options for the user. The first is to calculate the system Nash Equilibrium, which is done by applying the algorithm found in [31]. This value does not change with process architecture changes and the Nash values related to the local design variables of the currently selected designer are displayed in the top left of the DDT main window. The second operation is to simulate the response of the system to a step input, setting all design variables equal to *1*, and simulating the problem until it converges to within 2% the Nash solution as measured from the starting location. The same algorithm used in [43] is applied in this case. A less computationally intensive option which provides a generalized response time for the system can be found by checking *"Estimate Convergence Time"* which utilizes an approach adapted from control theory that estimates the settling time using the system eigenvalues. Further examining this approach in the context of distributed design is an area of current research. Finally, the user can examine the two largest system eigenvalues by checking the *"Show Eigenvalues"* option. The calculations for all these properties are provided through compiled MatLab© executables, called by the DDT as system applications by the DDT. Additionally, as the user makes changes to the system the analysis properties requiring updates to accurately reflect the new subsystem properties or process architecture are reset. In the next section several areas of future work are identified and discussed.

V. Conclusions

In this paper a toolkit, the Distributed Design Toolkit (DDT), is presented as a means for design managers to organize and analyze distributed design systems. The DDT operates on any XP or later Windows platform. In its current configuration, the DDT is capable of analyzing systems of up to twenty subsystems and twenty design variables. The analysis tools included in the DDT enable calculation of the Nash Solution, simulated convergence time, estimated convergence time and the system eigenvalues while providing users with a point and click GUI interface. The GUI commands associated with the DDT are modeled after those commonly seen in modern solid modeling packages to provide an intuitive interface through which users can manipulate system properties. In addition to the quantitative analysis tools included in the DDT, users are provided with access utilities to identify how the linkages between subsystems influence overall system behavior. These input and output connections between subsystems are shown as graphical connectors that identify the origin and destination of design process information.

Although the GUI and data handling is executed using C++ and OpenGL, a significant portion of the analysis is currently implemented through compiled MatLab© code. This functionality remains accessible to all users through the MatLab© compiler, but an important area of future work will focus on integrating these analysis tools in a solely C++ environment. Another key addition to the DDT will be increasing the allowable number of designers and design variables. To accomplish this, a streamlined approach is needed to enter objective function information and to accept analytical equations as objective function inputs. Incorporating these analytical expressions will require increased control over design variable names and a system to track these variables across multiple subsystems.

Another important area of future work is in developing techniques to analyze systems with nonlinear rational reaction sets. This analysis will leverage existing techniques using Lyapunov functions, but will also require advances in examining the solution process architecture. This task has two aspects: experimentally validating the existing approach to analyze solution process architecture and expanding its applicability to incorporate formulations for nonlinear rational reaction sets.

Acknowledgments

We would like to thank the National Science Foundation, grant DMII-0322783 for their support of this work and acknowledge the contribution of David Van Horn to this paper.

References

[1] Tappeta, R. V., and Renaud, J. E., 1997, "Multiobjective Collaborative Optimization," *ASME Journal of Mechanical Design*, 119(3), pp. 403-411.

[2] Gu, X., Renaud, J. E., Ashe, L. M., Batill, S. M., and Budhiraja, A. S., 2002, "Decision Based Collaborative Optimization," *ASME Journal of Mechanical Design*, 124(1), pp. 1-14.

[3] Wujek, B. A., Renaud, J. E., Batill, S. M., and Brockman, J. B., 1996, "Concurrent Subspace Optimization Using Design Variable Sharing in a Distributed Computing Environment," *Concurrent Engineering: Research and Applications*, 4(4), pp. 361-377.

[4] Giassi, A., Bennis, F., and Maisonneuve, J.-J., 2004, "Multidisciplinary Design Optimization and Robust Design Approaches Applied to Concurrent Design," *Structural and Multidisciplinary Optimization*, 125(1), pp. 124-130.

[5] Eiu, "Management Idea: Outsourcing," *The Economist*, 18 Mar 2009, pp. 13 – 19.

[6] Snyder, C. A., and Sankar, C. A., 1998, "Use of Information Technologies in the Process of Building the Boeing 777," *Journal of Information Technology Management*, IX(3), pp. 31-42.

[7] Tomayko, J. E., 2000, Computers Take Flight: A History of Nasa's Pioneering Digital Fly-by-Wire Project, United States Government Printing, NASA SP-2000-4224, Washington D.C.

[8] Rock, J., 2001, "Drive-by-Wire: The Next Step for the Auto Industry," *Weekly Corporate Growth Report*, January 22.

[9] Ming, X. G., Yan, J. Q., Lu, W. F., and Ma, D. Z., 2005, "Technology Solutions for Collaborative Product Lifecycle Management – Status Review and Future Trend," *Journal of Concurrent Engineering: Research and Applications*, 13(4), pp. 321-331.

[10] Atkins, D., Droegemeier, K. K., Feldman, C. I., Garcia-Molina, H., Klein, M. L., Messerschmitt, D. G., Messina, P., and Ostriker, J. P., 2003, "Revolutionizing Science and Engineering through Cyberinfrastructure," *Report of the National Science Foundation Blue Ribbon Panel on Cyberinfrastructure*.

[11] Siddique, Z., and Boddu, K., 2005, "A Cad Template Approach to Support Web-Based Customer Centric Product Design," *Journal of Computing and Information Science in Engineering*, 5(4), pp. 381-387.

[12] Leach, Y., Proulx, J., 2010, "Boeing Sets 787 First Delivery Date for Mid First Quarter 2011," *Boeing News Release*, 27 August 2010, Everett, WA.

[13] Gates, D., 2009, "Strike, Supplier Problems Cause Boeing to Deliver 100 Fewer Jets Last Year Than Expected," Seattle Times, January 9.

[14] Chanron, V., and Lewis, K., 2006, "A Study of Convergence in Decentralized Design Processes," *Journal of Research in Engineering Design*, 16(3), pp. 133-145.
[15] Devendorf, E., and Lewis, K., 2009, "Are We There Yet? Investigating the Role of Design Process Architecture on Convergence Rate," *ASME International Design Engineering Conference*, DETC2009-87517, San Diego, CA.
[16] Yi, S. I., Shin, J. K., and Park, G. J., 2007, "Comparison of MDO Methods with Mathematical Examples," *Journal of Structural and Multidisciplinary Optimization*, 35(5), pp. 391-402.
[17] Wieckem, M. M., and Reneke, J. A., 2005, "Complex Systems Decomposition under Uncertainty and Risk," *6th World Conference on Structural and Multidisciplinary Optimization* Rio de Janeiro, BR.
[18] Sobieszczanski-Sobieski, J., and Haftka, R. T., 1997, "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments," Structural and Multidisciplinary Optimization, 14(1), pp. 1-23.
[19] Allison, J., Kokkolaras, M., Zawislak, M., and Papalambros, P. Y., 2005, "On the Use of Analytical Target Cascading and Collaborative Optimization for Complex System Design," *6th World Congress on Structural and Multidisciplinary Optimization*, Rio de Janeiro, BR.
[20] Sobieszczanski-Sobieski, J., Agte, J. S., and Sandusky, R. R. J., 1998, "Bi-Level Integrated System Synthesis (Bliss)," NASA Paper No. 208715, Langley Research Center, Hampton, VA.
[21] Braun, R., 1996, "Collaborative Optimization: An Architecture for Large-Scale Distributed Design," Ph.D. Dissertation, Stanford University, Palo Alto, CA.
[22] Kim, H. M., Michelena, N. F., Papalambros, P. Y., and Jang, T., 2003, "Target Cascading in Optimal System Design," *ASME Journal of Mechanical Design*, 125(3), pp. 474-48.
[23] Eddy, J., 2006, "Solving Distributed, Non-Cooperative Design Problems Using Multi Agent Systems," Ph.D SUNY-Buffalo, Buffalo.
[24] Gurnani, A., Lewis, K.E., 2008, "Using Bounded Rationality to Improve Distributed Design," *AIAA Journal*, 46(12), pp. 3049-3059.
[25] Wujek, B., Renaud, J. E., and Batill, S., 1995, "A Concurrent Engineering Approach for Multidisciplinary Design in a Distributed Computing Environment," *Multidisciplinary Design Optimization: State of the Art*, N. Alexandrov, et al., eds., Hampton, VA, pp. 189-208.
[26] Owen, G., 1995, *Game Theory*, Academic Press, San Diego, CA.
[27] Rogers, J. L., 1989, "Demaid - a Design Manager's Aid for Intelligent Decomposition, User's Guide," TM101575, Langley Research Center, Hampton, VA.
[28] Rogers, J. L., 1996, "Demaid/Ga an Enhanced Design Manager's Aid for Intelligent Decomposition," *6th Annual AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA-96-4157, Seattle, WA.
[29] Vincent, T. L., 1983, "Game Theory as a Design Tool," *Journal of Mechanisms, Transmissions and Automation in Design*, 105, pp. 165-170.
[30] Nash, J., 1951, "Non-Cooperative Games," *The Annals of Mathematics* 54(2), pp. 286-295.
[31] Chanron, V., and Lewis, K., 2004, "Convergence and Stability in Distributed Design of Large Systems," *ASME International Design Engineering Technical Conferences*, DETC2004-57344, Montreal, Canada.
[32] Chen, C. T., 1999, *Linear System Theory and Design*, Oxford University Press, New York, NY.
[33] Chanron, V., and Lewis, K., 2003, "A Study of Convergence in Decentralized Design," *ASME International Design Engineering Technical Conferences*, DETC2003-48782, Chicago, IL.
[34] Chanron, V., Singh, T., and Lewis, K., 2004, "An Investigation of Equilibrium Stability in Decentralized Design Using Nonlinear Control Theory," *AIAA Multidisciplinary Analytical Optimization Conference*, AIAA-2004-4600, Albany, NY.
[35] Gopalakrishnan, J., Singh, T., and Lewis, K., 2006, "Enhanced Convergence in Distributed Design Processes," *ASME International Design Engineering Technical Conferences*, DETC2006-99544, Philadelphia, PA.
[36] Sosa, M., and Eppinger, S. D., 2003, "Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions," *ASME Journal of Mechanical Design*, 125(240), pp. 240-252.
[37] Smith, R. P., and Eppinger, S. D., 1997, "Identifying Controlling Features of Engineering Design," *Management Science*, 43(3), pp. 176-193.
[38] Sobieszczanski-Sobieski, J., 1990, "On the Sensitivity of Complex, Internally Coupled Systems," *AIAA Journal*, 28(1), pp. pp. 173-180.
[39] Rogers, J. L., and Bloebaum, C. L., 1994, "Ordering Design Tasks Based on Coupling Strengths," *5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA-94-4326, Panama City, FL.
[40] English, K., Bloebaum, C.L., and Miller, E., 2001, "Development of Multiple Cycle Coupling Suspension in the Optimization of Complex Systems," Structural and Multidisciplinary Optimization, 22(4), pp. 268-283, DOI: 10.1007/PL00013284.
[41] English, K. W., and Bloebaum, C. L., 2008, "Visual Dependency Structure Matrix for Multidisciplinary Design Optimization Tradeoff Studies," *Journal of Aerospace Computing, Information, and Communication*, 5(1), pp. 274-297.
[42] Devendorf, E., Devendorf, M., Lewis, K., 2010, "Using Network Theory to Model Distributed Design Systems," *AIAA Multidisciplinary Analytical Optimization Conference*, AIAA-2010, Fort Worth, TX.
[43] Devendorf, E,. Lewis, K., 2010, "Examining Interactions Between Solution Architecture and Designer Mistakes," *ASME International Design Engineering Technical Conferences*, DETC2010-29026, Montréal, Canada.