Multidisciplinary Environments: A History of Engineering Framework Development

Sharon L. Padula * and Ronnie E. Gillian †
NASA Langley Research Center, Hampton, Virginia, 23681

This paper traces the history of engineering frameworks and their use by Multidisciplinary Design Optimization (MDO) practitioners. The approach is to reference papers that have been presented at one of the ten previous Multidisciplinary Analysis and Optimization (MA&O) conferences. By limiting the search to MA&O papers, the authors can (1) identify the key ideas that led to general purpose MDO frameworks and (2) uncover roadblocks that delayed the development of these ideas. The authors make no attempt to assign credit for revolutionary ideas or to assign blame for missed opportunities. Rather, the goal is to trace the various threads of computer architecture and software framework research and to observe how these threads contributed to the commercial framework products available today.

Nomenclature

CAE	=	computer aided engineering
CDS	=	Conceptual Design Shop
DOE	=	Design of experiments
MA&O	=	Multidisciplinary Analysis and Optimization
MD	=	multiple discipline
MDO	=	multidisciplinary design optimization
MIMD	=	multiple input multiple data
OO	=	object-oriented
SD	=	single discipline
WWW	=	World Wide Web

I. Introduction

THIS paper traces the history of engineering frameworks and their use by MDO practitioners. For the purpose of this paper an engineering framework is defined as computer software designed to couple multidisciplinary analyses and to expedite the input, output and execution phases of those analyses. The history of engineering frameworks is far too large to be covered in a single conference paper. Therefore, we limited our study to papers that have been presented at one of the ten previous Multidisciplinary Analysis and Optimization conferences.

Our motivation for this study came from a NASA project called the Conceptual Design Shop (CDS). The CDS team surveyed commercial frameworks and compared their capabilities to a list of prioritized requirements[‡]. These requirements fall into the general categories of multidisciplinary analysis and optimization features, data management, user support and computer system compatibility as summarized in Fig 1. Several engineering frameworks including ModelCenter, a product of Phoenix Integration, Inc., iSIGHT and Fiper, products of Engineous Software, Inc., AML, a product of Technosoft, and Matlab, a product of The Mathworks, were rated against the requirements developed by the CDS team. All of the commercial frameworks received acceptable scores and the final choice of CDS framework was influenced by the prioritization assigned to each of the requirements by the disciplinary developers and the system analyst users of the CDS

* Senior Research Engineer, Aeronautics Systems Analysis Branch, Mail Stop 442, Associate Fellow AIAA.
† Branch Head, Advanced Engineering Environments Branch, Mail Stop 458.
‡ The use of trademarks or names of manufacturers in the report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

product.

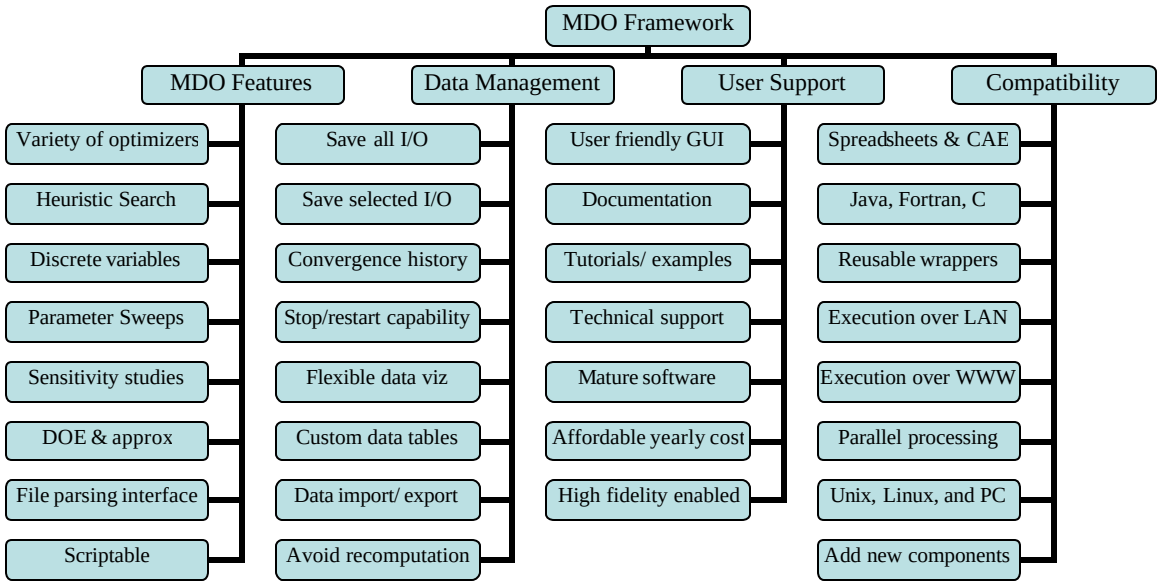

Figure 1. Desirable characteristics of MDO framework

The key features favored by the systems analyst users are described in this section in order to establish the current state of the art in MDO frameworks. These include a "plug and play" user interface, a selection of design exploration tools, and variable-fidelity reconfigurable models.

The term "plug and play" user interface implies that the user may choose from a selection of analysis codes. These codes may be easily linked and executed. The links may be easily broken, repaired and expanded. The execution sequence is adjusted based on input from the user and based on the outputs requested by the user. The codes may be written in a variety of programming languages and run on a distributed set of computers with a variety of operating systems. Popular commercial computer aided engineering (CAE) tools may be used in conjunction with research analysis codes. The tasks of code validation and verification, code maintenance, and code documentation can be separated from the task of multidisciplinary analysis and optimization.

The phrase "design exploration tools" denotes a selection of optimization, approximation, probabilistic analysis and data visualization tools. These tools can be easily selected and linked to a single analysis code or to an assembly of analysis codes. The tools should employ an intuitive set of menus and linking procedures so that the whole set of tools are available without extensive user training.

A variable-fidelity reconfigurable model implies that an assembly of linked codes and design exploration tools can be saved and reused at a later time. Any person reusing the model must be able to understand its configuration at a very abstract and at a very detailed level. The user must be able to delete some elements from the model and replace them with other elements that have a higher or a lower fidelity. The user must be able to replace some analysis codes with simple components such as table look-ups and equations which are easy to create inside of the MDO framework.

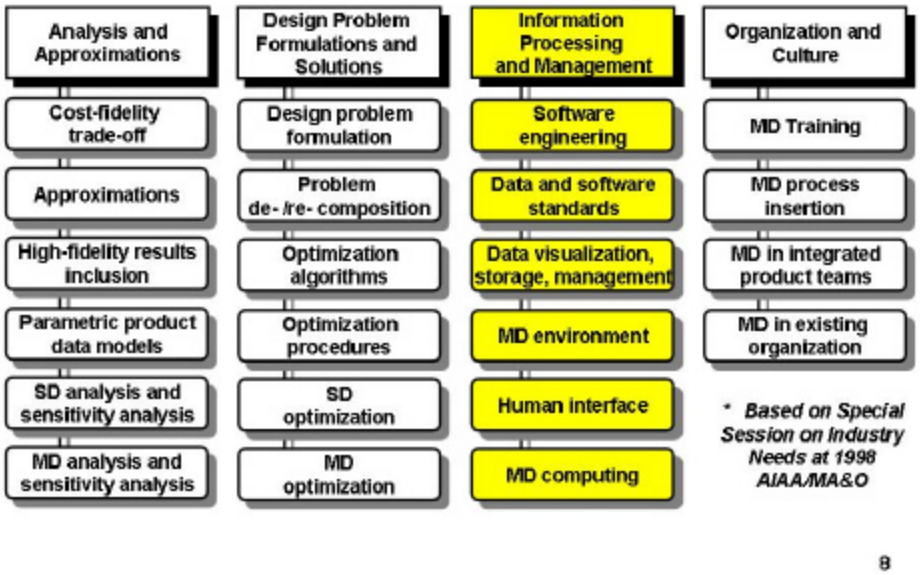

Figure 2. MDO elements as presented at the 7th MA&O Conference.

This paper begins with a general history of MDO framework research as it was reported at the MA&O conferences. This is followed by a history of several key features common in modern MDO frameworks.

II. History of MA&O Framework Research

Information processing and data management have always been elements of the MA&O conference. For example, Fig. 2 is adapted from a special session on Industry Needs at the 7th MA&O in 1998. Notice that software and computer framework issues are given equal importance with human factors, search algorithms and multidisciplinary analysis issues.

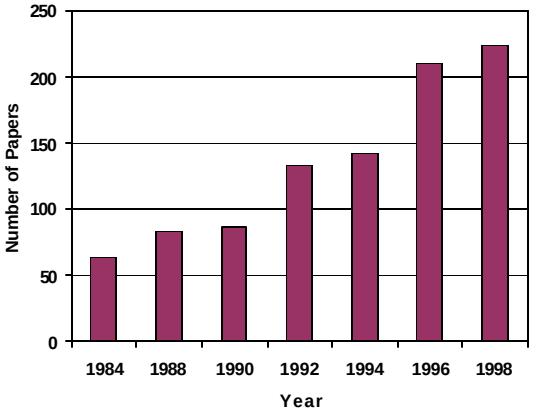

Figure 3. Number of papers published in the first seven MA&O Proceedings.

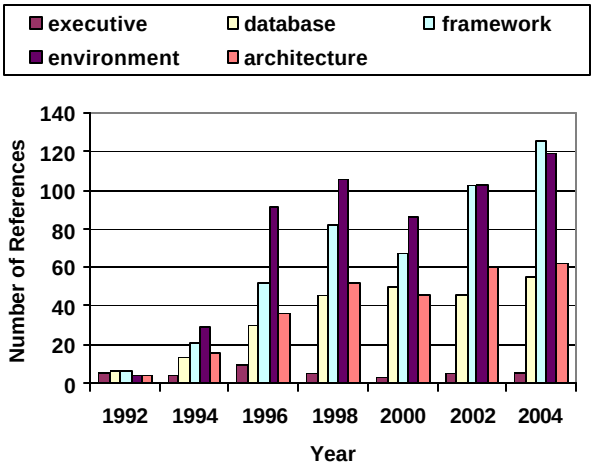

Figure 4. References to framework-related terms found via search of AIAA database.

As attendance at the MA&O conference grew, so did the number of papers that emphasized framework issues. Fig. 3 shows the number of papers published in the bound volumes of proceedings for the first seven conferences. At the

first three MA&O conferences, only about 10% of the papers involved information processing and data management issues. For example, 8 of 83 papers published in the 1988 proceedings discussed framework related topics. Even at the first AIAA sponsored MA&O conference in 1992, the number of papers emphasizing framework issues was still below 10%.

One way to observe the growth of framework research is to use the AIAA web-based report server. For example, Fig. 4 was created by searching for papers that included the word "multidisciplinary" in the conference name. That list of papers was filtered using key words such as "framework", "executive", and "environment". Even allowing for the fact that key words like framework can be used in contexts unrelated to information processing, Fig. 4 clearly indicates an increased number of references to framework topics beginning with the 6th MA&O in 1996.

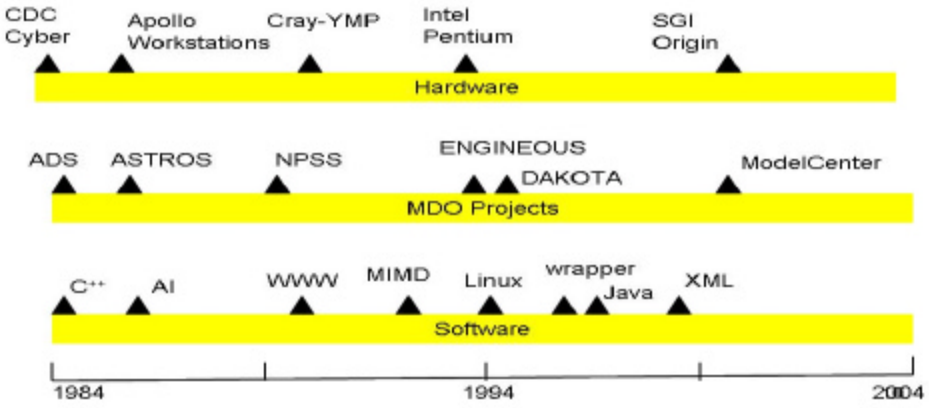

Figure 5. Timelines for hardware, software and MDO framework advances.

Obviously, the interest in MDO frameworks is correlated with advances in computer hardware and software. Fig. 5 captures this relationship in an informal manner. The figure presents three timelines indicating when the first mention of certain computer-related terms occurred in the MA&O proceedings. Papers published during the first ten years (i.e., 1984–1994) tend to mention monolithic codes written in procedural languages such as FORTRAN and C. These codes ran on super-computers like Cray–YMP or mini-computers like VAX–11/780 which are accessed via text-based terminals. During these same ten years, MDO frameworks, such as ASTROS[1], featured executive systems, executive control languages and special purpose databases. These early frameworks offered a fixed selection of specially modified analysis codes. It was quite difficult for the average researcher to add his own code to these frameworks. Papers published during the second decade (i.e., 1994–2004) mention a wider variety of framework architectures. These new options took advantage of hardware developments such as individual workstations, massively parallel processors, reliable random-access devices, fast networks and software developments such as the World Wide Web (WWW), object-oriented languages, script-based utilities and windows-based operating systems. The newer frameworks emphasize modularity and enable the coupling of analysis codes with little or no code modification.

III. Emergence of Key Ideas in Framework Research

The thesis of this paper is that key concepts arise long before they are truly practical. This idea is illustrated in framework research reported at the first ten MA&O conferences. Four key ideas are traced and typical references are cited. In many cases, the importance of these concepts is obvious today but was underestimated or misunderstood when the papers were presented. In other cases, ideas that were appropriate at a given time with the available hardware and software actually impeded the development of modern frameworks.

A. Modularity

Modularity is the first key concept. The term was originally applied to very complex computer codes that evolved from large, monolithic research codes. Just as individual codes are divided into subroutines to create more manageable and understandable software, so complex engineering application codes were architected to divide the system into smaller subsystems often referred to as modules. Modularity is used to isolate machine-dependent parts of the code, to improve the readability and to expedite testing. For example when the NASTRAN system was architected in 1965 the architects relied heavily on modularity to design, produce, and maintain their code[1]. The fact that the NASTRAN system architecture has lasted for decades with very little change to the original design proves this concept to be effective[2].

As MDO frameworks were developed, the concept of modularity meant that optimization tasks could be constructed from a library of interchangeable modules. The notion of modularity as applied to MDO frameworks is mentioned in each of the proceedings starting with the 1st MA&O. For example, Johnson and Venkayya observe that the MAPOL language ties together engineering analysis codes with a database and utilities library. This eliminates redundant coding and increases the reliability of the results[1]. In 1988, Campbell describes a system for aeroelastic analysis that was developed in a modular fashion so that more than one structures code and more than one aerodynamic code can be tested with a minimum of coding changes[3]. In both of these early examples, the linkage of modules to the executive is hard coded. An early example of a more flexible system is DYSCO which solves second order ODEs by assembling a model from a set of coupled components[4]. As early as 1992, the developers of RAMCOMP conclude that previous framework systems consisting of executive control plus a database are not effective. They propose that each engineering group must be responsible for configuration control, maintenance, documentation and training required for its own software while RAMCOMP provides the interface between software modules[5].

During the decade from 1996 to the present, various modular implementation schemes have been proposed and tested. In 1996, Hopkins and coworkers at NASA Lewis described a FORTRAN code called COMETBOARDS that implemented a "soft-coupling" between analysis and optimization routines[6]. Hopkins claims that several different optimization routines can be linked to any analysis which is tied to COMETBOARDS. Researchers at Sandia had a similar goal of isolating analysis from optimization when they developed a code called DAKOTA[7]. In this case, DAKOTA is an object-oriented code written in C++ such that each optimizer (e.g., NPSOL) is an instance of the optimizer base class which is itself an instance of a more general iterator base class. By using inheritance, each new optimization routine must only supply code for those functions that are unique to that method. In a 1998 paper, Engineous Software, Inc. announced the iSIGHT framework, the first commercial framework to incorporate an object-oriented optimization scheme[8]. The iSIGHT framework treated optimization, approximation and probabilistic analysis routines as interchangeable objects associated with data and with methods. This means that the iSIGHT framework is the first commercial MDO framework that can construct iterative tasks which include several levels of nested subtasks[9].

B. Data Handling

Database or data handling is the second key concept. Database software was first developed for use in the automation of business processes and included many concepts and computer constructs foreign to the engineering systems developer. Early engineering frameworks used card deck input (called runstreams) and output tapes or card decks plus a pool tape to collect intermediate data as the analysis progressed. The term database originally implied data storage including memory management and access control. For example, the Integrated Programs for Aerospace-Vehicle Design (IPAD) program developed the Relational Information Management System (RIM) data manager to address these problems in 1981[10]. The RIM code was one of the first systems that applied the principles of database management to engineering data.

Having a global database in standard format was a good way to enable efficient numerical analysis, and to encourage runstream reuse and execution restart capabilities. A central database in standard format also encourages multidisciplinary analysis because it reduces the number of translation routines needed. Each new analysis code must only be able to retrieve and store data from the database. Once the idea of a central database became conventional wisdom, then extensions to the database capabilities were developed. For example, data in standard format can be checked for consistency, completeness and adherence to user supplied upper and lower limits. Furthermore, data can be visualized using general purpose graphics, can be reasoned with using expert systems and can be printed using special purpose report generators[11].

Database technology for MDO was a significant research topic starting with the 2nd MA&O Conference. For example, Venkayya describes ASTROS and concludes that the development of software standards and the implementation of the CADDB database are some of the most significant contributions of that Air Force program[12].

In the same conference proceedings, researchers from the University of Iowa describe an object-oriented (OO) design methodology for databases. This OO method promises to deliver a product that performs well for its initial purpose and can be easily extended to unforeseen future products [13]. Many subsequent papers, such as one by Briggs in 1990, discuss the need for OO databases [14]. However, virtually all of these papers assume that the data will be organized around physical objects such as the structural components that make up an airplane and emphasize that the OO techniques are needed to provide multiple views of this data. Moreover, all three of the above papers assume that all data will be collected in a single central database.

A second set of database papers consider issues of maintainability and efficiency. For example, a Georgia Tech (GIT) study suggests that a central database approach has drawbacks as well as advantages. The GIT study favors an architecture where each module handles its own data because the specialized interfaces between a central database and other modules are a source of software maintenance costs [15]. The GIT study further suggests that only the inputs and outputs of each module need to be collected in a central database because these parameters can used by a "computational path generator" such as the one developed by Kroo at Stanford to decide which modules need to be executed and which can be skipped[16]. In 1990, Kroo elaborates on this idea and shows how quasi-procedural analysis can greatly reduce the computational effort for MDO by recalculating only the responses that have been invalidated by a given design variable permutation[17]. In 1998, Veley accomplishes a similar feat using the LISP-like language that comes with the AML framework[18]. Veley demonstrates that one significant advantage of the AML framework is this demand-driven execution of analysis modules.

A third set of database papers discuss the need for industrial strength databases which can enforce consistent models and implement company policy. For example, a 1988 paper by LTV describes the need for integration of all company databases. The LTV paper proposes a combined expert system plus database that can be used to enforce company procedures, collect physical and mechanical information, and estimate cost, maintainability and reliability of a design[11]. In 1994, Daum proposes that all application modules need to be linked to a database, not to each other[19]. Similarly in 1996, Rahn explains how CAD databases can be used to parametrically change the geometric configuration for structural optimization[20]. Samareh demonstrates an added advantage when CAD databases are used to enable multidisciplinary shape optimization. He recommends that each analysis and optimization module should process the NURBS files that are created by CAD software such as the Pro/Engineer software, a product of Parametric Technology Corporation[21]. Kingley takes this idea one step further by proposing an interpolation library so that a wide variety of CFD and FEM based analysis codes can exchange data with one another[22].

In 2004, Phoenix Integration announced a commercial product called CenterLink[23]. This paper makes the case for a much broader concept of a database. The CenterLink Analysis Library is a web-based database which organizes analysis models, input data and trade study outputs for every user with an account.

C. Parallel Processing

A third key concept is distributed computation or parallel processing. Originally, the idea was to automate multidisciplinary tasks even though each task required a special purpose computer processor. Later, the idea evolved to include the use of a large array of similar processors or individual work stations in order to reduce turnaround time. In both of these cases, the assignment of tasks to processors was not particularly flexible. Today, it is not unusual for an MDO framework to include load balancing and task assignment software so that the framework can decide where each task and module should be executed.

An early example of distributed computation is described by Brumbaugh and Duke in 1988[24]. The NASA Dryden Flight Research Facility connected a VAX minicomputer, UNIX workstations, and flight simulators using an Ethernet cable and TCP/IP protocol. This combination was necessary because no single computer had the computational speed, disk storage, FORTRAN, LISP and C compilers, high resolution graphics capabilities and flight simulation hardware to accomplish the real-time flight control studies. By 1990, most of the parallel processing research was focused on computer architecture and numerical algorithm compatibility issues. For example, the NASA Lewis Research Center (renamed Glenn Research Center) was developing propulsion system analysis codes such as NPSS to run on Intel iPSC 8 processor machines. Lytle and coauthors describe the ideal simulation environment as one which will use artificial intelligence to allocate parallel processors to single discipline and multidisciplinary analysis tasks as needed[25]. Whereas Lytle proposes parallel processing to make intractable computations possible, another author at the same conference proposes parallel processing to enable integrated product design. Rangan proposes a blackboard model which allows several designers to work at distributed workstations yet share ideas in a central location[26]. The parallel processing and AI research in the 1990 timeframe marks a dramatic shift in the way MDO was described. Prior to this an optimization process was described as a series of steps to be repeated until convergence. But parallel processing research encouraged

engineers to break the habit of serial thinking and to think of MDO as a set of tasks that could be completed in any order.

Although, many of the important parallel processing ideas existed in 1990, the implementation details were not reported until 1994 or 1996. For example, NASA Langley researchers describe a functioning parallel processing framework called FIDO that includes an executive control process, centralized data manager, graphical user interface and data monitoring capabilities[27]. At the same 5th MA&O conference, the important idea that a component is an analysis code plus a model that are wrapped together is proposed by Hale and Craig[28]. These authors explain that wrapped computer codes can contain additional information such as units and limitations and communicate with each other via predefined protocols. The power of this idea is demonstrated by the Boeing Support Services distributed software framework called "Access Manager"[29]. Ridlon explains that the key attributes of the Boeing framework are its OO design, coarse-grained dataflow and parallelization and its graphical user interface.

During the decade from 1996 to the present, both hardware and software advances encouraged the use of distributed processing. For example in 1996, Eldred modified the DAKOTA code to run on a distributed network of workstations and on a 256-node Intel Paragon[7]. That research compared running the analysis code in parallel on the Paragon versus running the gradient calculations in parallel on the workstations. Eldred reports that several versions of DAKOTA exist because some machines require a single program with multiple data while other machines allow multiple programs[30]. Rather than changing the code for each new machine, Becker and Bloebaum demonstrated distributed processing on a collection of workstations and personal computers using the Java language to create an architecture-neutral optimization framework[31]. Two years later, Allen and coauthors at NASA Ames describe a web-based framework for remote processing of planetary-entry vehicle design[32]. Users of the Ames framework can request runs and view results from any machine that has a web browser but all of the code remains in the original language and runs on the original computer. The real power of remote processing using Java was not realized for several years after the 1996 Becker prototype. By the 8th MA&O in 2000, Dovi describes using packages like CORBA and RMI to provide parallel processing utilities[33]. At the same conference, Alzubbi reviews the state of the art and discusses a framework called VADOR[34]. The VADOR framework allows wrapping of legacy codes, building of execution sequences from available components and execution using Java plus the RMI package. However, both Alzubbi and Dovi describe frameworks where the models are created by the system administrators and executed by the users. Truly flexible parallel processing under user control was delivered by commercial products such as the iSIGHT framework and the CenterLink framework[9,23].

D. User Interface

The final key concept is the user interface. Both graphical user interface (GUI) and method for multidisciplinary coupling are considered in this section. The original idea of a user interface was a text-based executive control language. Runstreams written in the executive control language provided a high-level summary of the job. These runstreams gave the user control over execution steps, and they collected input values plus output and execution specifications in one place. Moreover, the runstream could be saved, edited and reused in cases where the same job was rerun with minor changes. The next evolutionary development in user interfaces allowed the user to communicate with the executive system via pull-down menus and icons. Such a windowing system was easy to use and included the advantage of real-time data monitoring and execution control. Often, the underlying executive control language was preserved but the runstream was constructed automatically and could be saved and edited as before. More recently, the GUI was extended to provide "drag and drop" reconfiguration of modules, easy to use software wrapping facilities and more powerful utilities for optimization, analysis and data reporting. Typically, the newest systems produce executive control files which are not designed to be human readable and which must be modified using the GUI.

Each of the ten MA&O conferences included papers describing user interfaces for multidisciplinary coupling. At the early conferences, most MDO studies used monolithic FORTRAN codes with numeric input files. But many people understood the advantage of a control language and called for its development. Other authors added multidisciplinary modules to existing structural analysis codes like the Engineering Analysis Language (EAL) software, a product of EISI[35]. For example, at the 1st MA&O, Giles and Wrenn used the EAL executive and control language to enable a multilevel aircraft wing optimization[36]. The success of these early trials, led to the development of special purpose MDO languages such as SOL described by Lucas and Scotti[37] and MAPOL described by Neill[38]. The first example of an MDO language including an interactive user interface appears in 1988, when Berman describes linking of user supplied "components" for structural dynamics analysis[4]. At the same conference, several authors explained the value of windowing systems but most authors used the windows for editing command language input and visualizing results[11]. The exception to this rule was a paper by Hall and

Rogan. They experimented with HyperCard on a Macintosh computer to model the "design process as a network of interrelated decisions."[39] The idea was to construct an object-oriented language in which aircraft parts can be assembled and simple sizing and performance relationships can be evaluated. The beauty of this idea is that the data and analysis techniques are automatically associated with the selected object. Thus, some of the work of creating a model has been transferred from the user to the developer. It is important to note that developers Hall and Rogan thought that defining classes and attributes and methods so as to develop an intuitive and flexible user interface was a very challenging exercise.

By 1998, there were many competing ideas for user interfaces. Rogers and coauthors at Langley summarize the state of the art and provide a good reference list[40]. According to Rogers, each group that is working on user interfaces has a different idea about what the user needs. For example, Rogers proposes a web-based system for selecting multidisciplinary components plus an inference engine for controlling the order of execution. On the other hand, the developers of MDICE prefer a scripting language for execution control and a GUI to monitor progress and to associate codes with remote host machines[22]. As a third example, the developers of iSIGHT provide a GUI to monitor progress, to choose optimization methods and to build the scripting language[8].

In 2000, the FIPER team put some order into the discussion of user interface[41]. The team hypothesized that engineers need an adaptive workflow environment. The FIPER team report explains that the user creates an optimization task specification by selecting from available analysis codes, data sets, solution strategies, MDO methods, context, and sub-models. As work progresses, any and all of these selections are likely to change. Each of the twentieth century frameworks described in this paper allowed changes to one or two of these optimization task specifications but no framework that existed in 2000 had the flexibility to change all of the specifications.

The period from 2000 to the present was the most active period for improvement to user interface. For the first time, the average user of the framework had tools to wrap their own analysis codes and configure their optimization tasks and subtasks without needing sophisticated computer programming knowledge. For example, the FIPER team developed a Windows-based system for wrapping analysis codes and making them available from a user library[42]. And in Europe, a team from NLR described a similar framework called SPINEware that used Java applets and a web browser[43]. Both FIPER and SPINEware were especially suited for integrating software and distributing tasks between design teams in physically distance locations. Meanwhile, a team from Lockheed used the AML commercial software as a framework for building and optimizing variable fidelity CAD models with parameterized geometry[44]. This framework used the idea of an object-oriented database to build consistent variable fidelity models and used the idea of data dependencies to avoid recalculation of expensive CFD and FEM analysis. A team from GM built a similar capability using iSIGHT plus and an integrated CAD and database package[45].

IV. Concluding Remarks

It is fascinating to look at the history of MDO framework development and to see how ideas arise, get expanded, and eventually get replaced by better ideas. One purpose of this paper is to recognize and record the many small advances that contribute to a major new computing capability. Another purpose of this paper is to understand how conventional wisdom can get in the way of new developments. A final purpose is to enumerate the roadblocks to new framework technology.

This paper shows that several of the key advances in MDO frameworks occurred 15- 20 years ago. The idea of modularity, where engineering codes need to be separated from machine-dependent utilities, is the oldest of the four key ideas. An important enabler of modularity is the identification of computational tasks that can be done by general purpose code rather than by special purpose code. Databases, parallel processing utilities, MDO methods and user interfaces are all examples of general purpose modules that have been developed and used by the MDO community.

Several ideas that were popular 20 years ago are seldom mentioned today. For example, at one time everyone assumed that an MDO framework would consist of an executive controller, a central database and a text-based user interface on a single supercomputer. Today, researchers favor a web-based interface with a very limited central controller, federated components and greatly restricted data transfer. In these new frameworks, the user has control over the selection of modules, the execution of the process and the post-processing of the data. Another discredited idea is that computationally expensive MDO procedures must use a monolithic code for efficiency. This idea was problematic because the existing analysis codes tend to be written in more than one computer language and the process of integrating several codes is time consuming, introduces bugs and leads to a huge number of versions of each code. A final discredited idea is that treating an aircraft as a collection of components and using C^{++} means that one is practicing object-oriented programming techniques. We now understand that OO codes require careful

planning and proper allocation of functions to classes. Only then will good user interfaces and efficient, maintainable code result.

This paper proposes that the key ideas used to create commercial frameworks such as the SIGHT and ModelCenter are found in 15 year old MA&O papers. It is interesting to speculate why these ideas did not bear fruit sooner. One idea introduced above is that MDO frameworks were delayed because researchers were conditioned to think of design and optimization as a step by step process that could be sufficiently captured in a text-based runstream. This mindset made it difficult to think of MDO as a set of client/server tasks which can be activated as needed by a user or by an optimization strategy. Another idea is that general purpose frameworks were delayed because no organization was able or willing to pay the development cost. As observed in Ref. 39 by Hall and Rogan, "coming up with a good design process is basically the same as coming up with a good design." Over and over again, MDO researchers elected to come up with a good point design rather than a good framework. Unfortunately, when a good design was produced, the process of producing that design was often lost or discarded before it was required by the next design cycle or even the next level of re-work.

Anyone who looks through all the papers in the References section will surely conclude that the history of framework development was greatly influenced by rapid changes in computer hardware and software. We believe that the high cost of computers, memory, storage and networks, plus the large number of unique operating systems and the lack of standardized software utilities delayed the development of general purpose MDO frameworks. Moreover, we believe that continued improvements in hardware and software will influence future framework development and will make the deficiencies in current framework systems even more apparent. It is interesting to speculate what the next generation of MDO frameworks will look like when computers can compute faster than engineers can think. We predict that future frameworks will face challenges in modularity, data handling, parallel processing and user interfaces. For example, one key issue in modularity will be verification and validation of MD systems constructed with federated modules. And a key issue in data handling will be mining enormous quantities of data for design knowledge. For parallel processing, the issue will always be the speed of data transfer over the network and for user interface the issues of configuration management and process visualization will be important.

References

[1]Johnson, E. H., and Venkayya, V. B., "Progress Report on the Automated Strength-Aeroelastic Design of Aerospace Structures Program," *Recent Experiences in Multidisciplinary Analysis and Optimization*, NASA CP–2327, 1984, pp. 527–538.

[2]Johnson, E. H., "Considerations in the Performance of Large Structural Optimization Tasks, *7th AIAA Multidisciplinary Analysis and Optimization*, CP9811, AIAA, Reston, VA, 1998, pp. 165–169, AIAA-1998-4728.

[3]Campbell, Richard L, "Calculated Effects of Varying Reynolds Number and Dynamic Pressure on Flexible Wings at Transonic Speeds," *Recent Experiences in Multidisciplinary Analysis and Optimization*, NASA CP–2327, 1984, pp 309–327.

[4]Berman, Alex, "A Generalized Software Executive for Multidisciplinary Computational Structural Dynamics," *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP–3031, 1988, pp. 545–563.

[5]Volk, J. A., "Multidisciplinary Design Environment Development for Air Vehicle Engineering," *4th AIAA Multidisciplinary Analysis and Optimization*, CP9213, AIAA, Washington, DC, 1992, pp. 1558–1567, AIAA–92–1113.

[6]Hopkins, D. A., Patnaik, S. N., Berke, L, "General-purpose Optimization Engine for Multidisciplinary Design Applications," *6th AIAA Multidisciplinary Analysis and Optimization*, CP9611, AIAA, Reston, VA, 1996, pp. 1558–1567, AIAA–1996–4163.

[7]Eldred, M. S., Hart, W. E., Bohnhoff, W. J., Romero, V. J., Hutchinson, S. A., Salinger, A. G., "Utilizing Object-Oriented Design to Build Advanced Optimization Strategies with Generic Implementation," *6th AIAA Multidisciplinary Analysis and Optimization*, CP9611, AIAA, Reston, VA, 1996, pp. 1568–1582, AIAA–1996–4164.

[8]Golovidov, Oleg, Kodiyalam, Srinivas, Marineau, Peter, Wang, Liping, and Rohl, Peter, "Flexible Implementation of Approximate Concepts in an MDO Framework," *7th AIAA Multidisciplinary Analysis and Optimization*, CP9811, AIAA, Reston, VA, 1998, pp. 1969–1979, AIAA–1998–4959.

[9]Koch, Patrick N., Wujek, Brett, and Golovidov, Oleg, "A Multi-Stage Parallel Implementation of Probabilistic Design Optimization in an MDO Framework," *8th AIAA Multidisciplinary Analysis and Optimization Papers on Disc* [CD-ROM], AIAA, Reston, VA, 2000, AIAA–2000–4805.

[10]Boeing Computer Service, "BCS RIM – Relational Information Management System Version 6.0 User Guide," Boeing Computer Service Company, Seattle, WA, 1983.

[11]Green, T. L., Jordan, B. M., and Oglesby, T. L., "The Designer of the 90's: A Live Demonstration," *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP–3031, 1988, pp. 373–385.

[12]Venkayya, V. B., "Recent Developments in Large-Scale Structural Optimization," *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP–3031, 1988, pp. 1521–1540.

[13]Dopker, Bernhard, and Haug, Edward, "A Large Scale Software System for Simulation and Design Optimization of Mechanical Systems," *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP–3031, 1988, pp. 1319–1333.

[14]Briggs, H. C., "Control/Structure Interaction Conceptual Design Tool," *Proceedings of 3rd Multidisciplinary Analysis and Optimization*, 1990, pp. 178–184.

[15]Bates, P. R., and Schrage, D. P., "TRUSS: An Intelligent Design System for Aircraft Wings," *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP–3031, 1988, pp. 333–355.

[16]Kroo, I. and Takai, M., "A Quasi-Procedural, Knowledge-Based System for Aircraft Design," AIAA 88–4428, 1988.

[17]Kroo, Ilan and Takai, M., "Aircraft Design Optimization using Quasi-Procedural Method and Expert System," *Proceedings of 3rd Multidisciplinary Analysis and Optimization*, 1990, pp. 464–469.

[18]Veley, Duane E. "Optimization in an Adaptive Modeling Language, *7th AIAA Multidisciplinary Analysis and Optimization*, CP9811, AIAA, Washington, DC, 1998, pp. 1248-1255, AIAA–1998–4872.

[19]Daum, A. and Wolf, D. M., "TRANSYS – A Multidisciplinary Software System for Preliminary Design, Analysis, and Evaluation of Space Transportation Systems," *5th AIAA Multidisciplinary Analysis and Optimization*, CP9413, AIAA, Washington, DC, 1994, pp. 860–869, AIAA–1994–4342.

[20]Rahn, M., Schoettle, U. M., and Messerschmid, E., "Multidisciplinary Design Tool for System and Mission Optimization of Launch Vehicles," *6th AIAA Multidisciplinary Analysis and Optimization*, CP9611, AIAA, Reston, VA, 1996, pp. 1266–1274, AIAA–1996–4130.

[21]Samareh, Jamshid A., "Use of CAD Geometry in MDO," *6th AIAA Multidisciplinary Analysis and Optimization*, CP9611, AIAA, Reston, VA, 1996, pp. 88–98, AIAA-1996-3991.

[22]Kingsley, Gerry, Siegel, John M., Harrand, Vincent J., Lawrence, Charles, and Luker, Joel J., "Development of a Multidisciplinary Computing Environment (MDICE)," *7th AIAA Multidisciplinary Analysis and Optimization*, CP9811, AIAA, Reston, VA, 1998, pp. 251–260, AIAA–1998–4738.

[23]Woyak, Scott, Kim, H., and Mullins, J., "A Web-Centric Architecture for Deploying Multidisciplinary Engineering Design Processes," *10th AIAA Multidisciplinary Analysis and Optimization*, AIAA, Reston, VA, 2004, AIAA–2004–4498.

[24]Brumbaugh, R. W., and Duke, E. L., "Real-Time Application of Knowledge-Based Systems," *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP–3031, 1988, pp. 357–371.

[25]Lytle, J. K., Remaklus, D. A., and Nichols, L. D., "Numerical Propulsion System Simulation," *Proceedings of 3rd Multidisciplinary Analysis and Optimization*, 1990, pp. 115–128.

[26]Rangan, R. M., and Fulton, R. E., "An Information Driven Strategy to Support Multidisciplinary Design," *Proceedings of 3rd Multidisciplinary Analysis and Optimization*, 1990, pp. 457–463.

[27]Weston, R. P., Townsend, J. C., Eidson, T. M., and Gates, R. L., "A Distributed Computing Environment for Multidisciplinary Design," *5th AIAA Multidisciplinary Analysis and Optimization*, CP9413, AIAA, Washington, DC, 1994, pp. 1091–1097, AIAA–94–4372.

[28]Hale, M. A. and Craig, J. I., "Preliminary Development of Agent Technologies for a Design Integration Framework," *5th AIAA Multidisciplinary Analysis and Optimization*, CP9413, AIAA, Washington, DC, 1994, pp. 413–422, AIAA–94–4297.

[29]Ridlon, Stephen A., "A Software Framework for Enabling Multidisciplinary Analysis and Optimization," *6th AIAA Multidisciplinary Analysis and Optimization*, CP9611, AIAA, Reston, VA, 1996, pp. 1280–1285, AIAA–96–4133.

[30]Eldred, M. S. and Hart, W. E., "Design and Implementation of Multilevel Parallel Optimization on the Intel Teraflops," *7th AIAA Multidisciplinary Analysis and Optimization*, CP9811, AIAA, Reston, VA, 1998, pp. 44–54, AIAA–1998–4707.

[31]Becker, J. C., and Bloebaum, C. L., "Distributed Computing for Multidisciplinary Design Optimization using Java," *6th AIAA Multidisciplinary Analysis and Optimization*, CP9611, AIAA, Reston, VA, 1996, pp. 1583–1593, AIAA–1996–4165.

[32]Allen, G. A., Gage, P. J., and Venkatapathy, E., "A Web-based Analysis System for Planetary Entry Vehicle Design," *7th AIAA Multidisciplinary Analysis and Optimization*, CP9811, AIAA, Reston, VA, 1998, AIAA–1998–4826.

[33]Dovi, Augustine R., Su, Phillip, and Murthy, Tam, "A Distributed Object Approach to Multidisciplinary Analysis of Aerospace Vehicles," *8th AIAA Multidisciplinary Analysis and Optimization Papers on Disc* [CD-ROM], AIAA, Reston, VA, 2000, AIAA–2000–4898.

[34]Alzubbi, Abdulsalam, Ndiaye, Amadou, Mahdavi, Babak, Guibault, Francois, Ozell, Benoit, and Trepanier, Jean-Yves, *8th AIAA Multidisciplinary Analysis and Optimization Papers on Disc* [CD-ROM], AIAA, Reston, VA, 2000, AIAA–2000–4903.

[35]Whetstone, W. D., "EISI-EAL Engineering Analysis Language Reference Manual," Engineering Information Systems, Inc. July 1983.

[36]Giles, G. L., and Wrenn, G. A., "Multidisciplinary Optimization Applied to Transport Wing," *Recent Experiences in Multidisciplinary Analysis and Optimization*, NASA CP–2327, 1984, pp. 439–453.

[37]Lucas, S. H., and Scotti, S. J., "The Sizing and Optimization Language (SOL)- A Computer Language to Improve the User/Optimizer Interface," *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP–3031, 1988, pp. 601–619.

[38]Neill, D. J., "ASTROS – A Multidisciplinary Automated Structural Design Tool," *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP–3031, 1988, pp. 529–543.

[39]Hall, D. W., and Rogan, J. E., "Development of a Micro-Computer Based Integrated Design System for High Altitude Long Endurance Aircraft," *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP–3031, 1988, pp. 275–296.

[40]Rogers, James L., Salas, Andrea O., and Weston, Robert P., "A Web-based Monitoring System for Multidisciplinary Design Projects," *7th AIAA Multidisciplinary Analysis and Optimization*, CP9811, AIAA, Reston, VA, 1998, pp. 35–43, AIAA–1998–4706.

[41]Wujek, Brett, Koch, Patrick N., and Chiang, Wei-Shan, "A Workflow Paradigm for Flexible Design Process Configuration in FIPER," *8th AIAA Multidisciplinary Analysis and Optimization Papers on Disc* [CD-ROM], AIAA, Reston, VA, 2000, AIAA–2000–4868.

[42]Wujek, B., Koch, P., McMillan, M., Chiang, W., "A Distributed Component-Based Integration Environment for Multidisciplinary Optimal and Quality Design," *9th AIAA Multidisciplinary Analysis and Optimization*, AIAA, Reston, VA, 2002, pp. 716–723, AIAA–2002–5476.

[43]Vankan, W., and Laban, M., "A SPINEware-based Computational Design Engine for Integrated Multidisciplinary Aircraft Design," *9th AIAA Multidisciplinary Analysis and Optimization*, AIAA, Reston, VA, 2002, pp. 431–441, AIAA–2002–5445.

[44]Ahlqvist, A., Nayfeh, J. F., Kodiyalam, S., and Zarda, P.R., "Object-oriented Multidisciplinary Design Optimization," *8th AIAA Multidisciplinary Analysis and Optimization Papers on Disc* [CD-ROM], AIAA, Reston, VA, 2000, AIAA–2000–4784.

[45]Fenyes, P., Donndelinger, J. and Bourassa, J., "A New System for Multidisciplinary Analysis and Optimization of Vehicle Architectures," *9th AIAA Multidisciplinary Analysis and Optimization*, AIAA, Reston, VA, 2002, pp. 1020–1028, AIAA–2002–5509.