Advances in Engineering Software 40 (2009) 176–183

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Development of Web services-based Multidisciplinary Design Optimization framework

Ho-Jun Lee, Jae-Woo Lee, Jeong-Oog Lee *

Department of Aerospace Information Engineering, Konkuk University, 1 Hwayang-dong, Gwangjin-Gu, Seoul 143-701, South Korea

ARTICLE INFO

Article history:
Received 24 August 2007
Received in revised form 14 February 2008
Accepted 14 March 2008
Available online 22 May 2008

Keywords:
Multidisciplinary Design Optimization
Web services
Globus Toolkit
Workflow management system
Agent

ABSTRACT

The defining characteristic of a Multidisciplinary Design Optimization (MDO) strategy or method, compared to the more traditional, sequential approach to conducting design work, is that the contributions of all mutually influential disciplines are concurrently taken into account. Therefore, a framework that allows the implementation of MDO methods must be an environment for design synthesis. It is also desired that the user of an MDO framework be capable of efficiently integrating and managing the resources distributed over heterogeneous platforms. This paper proposes a Web services-based MDO framework that enables the synthesis of available disciplinary and cross-disciplinary resources for MDO via the Globus Toolkit. Examples of organic and autonomous execution of MDO methods are presented to highlight the effectiveness of modern automation techniques, such as workflow management system and agent technology. The salient features of a planned collaborative design environment, which will be built through Web-based user interfaces, are discussed last.

1. Introduction

Engineering design is inherently a multi-step process. Taking the aerospace domain as an example, all design-related activities fall under one of the major phases that are followed in the order of conceptual design, preliminary design, detailed design, manufacturing, production, and operations and support. A well-known deficiency of such a traditional approach to product development is that a large portion of the system's life-cycle cost and quality become fixed by the end of the early design phases, during which the contribution from each discipline is notably uneven [1]. The realization that front-loading the design cycle with increased cross-disciplinary trade studies could enable the elicitation of more decision-critical information, and thus favorably impact the performance, cost, quality, and various – illities (e.g., reliability) of the product, is one of the key motivations for the push towards Multidisciplinary Design Optimization (MDO) [2].

The defining characteristic of an MDO strategy or method, compared to the more traditional and sequential approach to conducting design work, is that the contributions of all mutually influential disciplines are concurrently taken into account. Implementing an MDO method naturally facilitates the involvement of disciplinary experts that would have otherwise been difficult to initiate during the earliest stages of product development. As if to demonstrate this point, MDO has been embraced early by the aerospace and automotive sectors. Both air and ground vehicles represent systems that are so complex, in both the scope and number of interdependencies between all involved disciplines, that product competitiveness cannot be achieved without multidisciplinary collaboration. Because the end-goal of an MDO method is to identify an optimum design solution which either satisfies or exceeds customer expectations, the applicability of MDO appears to have extended to fast-paced industries such as the electronics industry.

MDO methods come in many different guises, ranging from the traditional calculus-based numerical optimization techniques [3] to more recent developments such as evolutionary algorithms [4], multi-attribute optimization [5], and reliability-based optimization [6]. If properly formulated and correctly implemented, MDO strategies can allow the shortening of design cycle times, reduction of engineering and development costs, and improvement of product quality, all of which are causative factors for increasing the profit, productivity, or competitiveness of an organization. The missing key to multidisciplinary success is therefore the existence of a software environment for design synthesis; that is, an MDO framework.

At a minimum, a functional MDO framework must allow the integration of disciplinary analysis codes as well as tools for facilitating cross-disciplinary tradeoffs. The larger the organization, the more likely it is that the required disciplinary and inter-disciplinary resources exist in a heterogeneous computing environment. Even for small to moderate sized institutions, it is unlikely that

* Corresponding author. Tel.: +82 2 450 3435; fax: +82 2 444 6670.
E-mail address: ljo7@konkuk.ac.kr (J.-O. Lee).

0965-9978/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2008.03.015

all of their homegrown and commercial simulation codes are centrally located on a bank of homogenous workstations. As a matter of fact, the computing environment of the future is projected to be further dependent and based on the Web, due to the growing popularity of the "ubiquitous network" concept [7].

It is thus apparent that a need exists for a modern MDO framework that not only enables the synthesis of multidisciplinary analysis and optimization activities, but also the creation of a Virtual Organization (VO), a collection of entities that share the computing resources for a common purpose [8]. Motivated to address this technological gap, the authors present a novel Web services-based MDO framework that was developed using the Globus Toolkit Ver. 4 (GT4). GT4 is a middleware for building computing grids and provides a number of built-in services that conform to the Open Grid Forum (OGF) protocols [9].

In order to retain the generality of the framework, it must be capable of integrating any mix of analysis and optimization codes as per the design problem at hand. The obvious challenge here is that, with the exception of a few popular commercial software packages (e.g., CATIA, VisualBasic, Matlab, etc.), most computer programs are originally written to be standalone versions native to specific computing platforms. Wrapper and parser technologies are adopted accordingly to enable seamless communication between the integrated resources, regardless of their geographical location and/or native operating system requirements. Specifically, a form of eXtensible Markup Language (XML) is invoked to ensure that the data shared among two or more integrated tools are in a mutually readable format. In addition, agent technology is utilized for the dual purpose of workflow and data management. This paper presents examples of agent, wrapper and parser models developed for our MDO framework. The end-result is that an organic and autonomous execution of a desired MDO strategy of any scale becomes a possibility, with an added advantage of being able to control it from a homogenous, single-user environment.

Lastly, an ideal MDO framework would double as a collaborative design environment where the participation and interaction of disciplinary specialists can guide the overall multidisciplinary analysis, synthesis, and optimization activities. A powerful Web-based user interface system was thereby made to be a feature of our framework in order to allow the solicitation of expert disciplinary knowledge at anytime and from anywhere.

2. MDO framework

2.1. Background

In his seminal 1993 paper, Sobieski defines MDO as "a methodology for design of complex engineering systems that are governed by mutually interacting physical phenomena and made up of distinct interacting sub-systems" [10]. The terms Multidisciplinary Analysis and Optimization (MAO), Multidisciplinary Design Methodology (MDM), and Multidisciplinary Design Technology (MDT) have also been reportedly used to describe this then emerging field of engineering. In the aerospace community, however, the establishment of the AIAA Technical Committee on Multidisciplinary Design Optimization (TC-MDO) in 1991 [1] has led to the acknowledgement and subsequent wide-spread usage of the acronym.

The principal "conceptual components" of MDO, according to Sobieski, are Design-Oriented Analysis, Approximation Concepts, System Mathematical Modeling, Decomposition, Design Space Search, Optimization Procedures, and Human Interface. Over the years, active research into MDO methods has produced a large body of strategies that accentuate one or more of the above aspects of MDO. For example, early efforts focused on the Design-Oriented Analysis aspect of MDO, culminating in the formulation of the System Sensitivity Analysis (SSA) technique [3]. Approximation Concepts refer to a class of methods that leverage upon statistical data modeling techniques, also known as surrogate modeling or parameter design methods in the literature [11]. Surrogate models are usually closed-form algebraic equations, which enable greatly expedited evaluations of the underlying physical phenomena or analysis codes. The application of methods based on Design of Experiments theory, such as Taguchi orthogonal arrays, Response Surface Method (RSM) and Variable Complexity Method, to aircraft and spacecraft design work are numerous and well documented. Recent investigations into advanced pattern-matching (interpolation) techniques such as Kriging [12] and Neural Networks have stimulated an increasing interest in adapting them for large-scale aerospace systems design problems [13]. The Design Space Search aspect of MDO has been largely addressed by the adoption of traditional, gradient-based search algorithms as well as more recent exploratory algorithms, such as Genetic Algorithm (GA), Simulated Annealing (SA), and Ant Colony Optimization (ACO). Both Multidisciplinary Feasible (MDF) and Individual Discipline Feasible (IDF) methods can be categorized as strategies aimed at addressing Decomposition. Lastly, Collaborative Optimization (CO) [14] and Bi-Level Integrated System Synthesis (BLISS) [15] are examples of MDO methods that deal with both multi-level Decomposition and Optimization Procedure. Lastly, the reported applications of expert system (e.g., artificial intelligence) tools to aircraft design problems are precedents of emphasizing the Human Interface aspect of MDO [16].

Because the underlying philosophy of MDO is the synergistic integration of all contributing disciplines in finding a system-level optimum, MDO methods are especially applicable to the design of systems with highly interdependent sub-systems. In the aerospace domain, vehicle platforms (aircraft, rotorcraft, spacecraft, space launch vehicles, etc.) are good representations of systems whose performance is sensitive to cross-disciplinary trades. Taking a typical aircraft design problem as an example, an MDO approach would work well in reconciling the challenges posed by inter-disciplinary couplings; i.e., the interactions between aerodynamics, propulsion, structures, controls, and mission analysis. The application of a well-structured MDO method would thus enable the identification of a balanced multidisciplinary design in terms of the most significant system-level metrics; e.g., take-off gross weight, mission capabilities, operating cost, etc.

The payoffs of implementing MDO in the design of an engineering system can be numerous. As previously mentioned, a well planned and properly executed MDO method can reduce a product's design and development cycle times and costs while improving the – illities. Ref. [17] is a documentation of some of the earliest success stories of MDO in the context of aircraft design. Since then numerous reports of MDO-related work can be found in the general technical literature.

2.2. Necessity of an MDO framework

MDO, while noble in principle, is nonetheless a non-trivial task to carry out in practice. This is largely due to the nature of a multidisciplinary, complex engineering system. It is usually the case that a single monolithic model representing the entirety of the system architecture is unavailable. What a systems engineer or an MDO specialist often encounters is a fragmented assembly of sub-models, each embodying a particular discipline, physical hardware component, or design phase. These sub-models are also interdependent on one another, meaning that the outputs of a disciplinary analysis become the inputs to one or more other analyses, and consequently leading to design iterations. For an enterprise-level implementation of MDO, the issue is further complicated by the fact that the sub-models are: owned and oper-

ated by different professionals; associated with specific computing platforms; reside in geographically dispersed locations; or all of the above. Therefore, a manual and non-automated execution of an MDO strategy is likely to be prone to numerous human errors, not to mention the extra time and cost it would impose on the entire design process, thereby negating the advantages of practicing MDO in the first place.

On the contrary, a software framework that allows the creation of an integrated design environment, which can also be automated as per its intended usage, would be the key to realizing the many benefits of implementing MDO methods. Such a framework, which is henceforth referred to as the MDO framework, would reduce both time and effort spent on ensuring the compatibility of data formats between the connected sub-models (e.g., an array is passed on to a correctly sized array, string variable is not erroneously associated with the data-type double, units are consistent, etc.). A highly capable MDO framework could also serve as the backbone of an interactive design environment that can constantly evolve as the collaboration amongst the involved disciplinary experts increase with design progression.

The resources that are available to an MDO framework can largely be classified into two categories: disciplinary and cross-disciplinary resources. The former encompasses simulation-oriented elements such as disciplinary analysis codes, Computer-Aided Design (CAD) tools, etc. while the latter includes programs that can be accessed for inter-disciplinary trade studies or shared among all involved disciplinarians: e.g., optimization algorithms, Database Management System (DBMS), etc.

Code execution within an MDO framework can occur in one of the following three ways. The simplest of the three is the individual execution of a standalone code which is viewed as a black box; i.e., in terms of its input and output characteristics. An example of this would be the running of a statistical atmospheric model that, when supplied with altitude, outputs values for air pressure, temperature, density, speed of sound, etc. Second, multiple disciplinary codes can be organically linked to one another to represent a multidisciplinary analysis routine. The cross-disciplinary relationships between the codes determine whether they are to be executed serially, parallelly, or iteratively. Computationally, the interdependencies between the codes are defined in terms of their execution sequence, start and stop criteria, and input/output variables (data). Lastly, an optimization algorithm can be coupled with either single disciplinary analysis or multidisciplinary analysis routines. Due to the nature of numerical optimization, this third case is almost always an iterative scheme.

Generally, it is not guaranteed that the above-mentioned resources exist in a homogenous design environment. More often than not, the necessary codes are found distributed through multiple machines and across a heterogeneous computing environment, which can turn code integration into a time-consuming and menial drudgery. An MDO framework should thus provide a systems integrator with a host of automation techniques that allows the simplification of design synthesis, convenient access to disciplinary and cross-disciplinary resources, and straightforward implementation of various MDO methods. In order to develop such a functional MDO framework, a broad range of computing technologies is required. What follows is a brief discussion on the most critical of such requirements.

2.3. Requirements for building a functional MDO framework

Typically, an MDO framework can be generalized to consist of the following elements: an intuitive and easy-to-use Graphical User Interface (GUI); CAD tools for configuration design; DBMS for data storage and management; and both disciplinary (e.g., analysis codes) and cross-disciplinary (e.g., optimizers) resources which are also the core elements of a given framework's infrastructure. This section first reviews the development aspects of the requirements for building a useful MDO framework, followed by a discussion on the same requirements from a functional perspective.

Above all, the GUI is one of the fundamental requirements in developing a MDO framework. Possible uses of the GUI include, but are not limited to: integration of CAD and CAE tools; assemblage and management of data flow paths between integrated analysis codes; optimizer control; enhancement of MDO problem definition. Additional requirements to be considered during the framework's development are: compatibility with a heterogeneous computing environment; centralized DBMS; a debugger for identifying erroneous analysis and invalid process flow paths; capability to monitor the progression history of design parameter values; real-time deletion, addition, and modification of process elements; the ability to handle large-scale problems with up to hundreds of design variables; and inexpensive reanalysis (e.g., the ability to resume analysis from a previous design point that had failed). An object-oriented structure and compliance with open development standards are essential to ensure the future extensibility of the framework software. Lastly, conformity to the Workflow Management System (detailed in Section 4) and parallel execution of wrapped analysis codes are examples of non-essential yet supplementary requirements.

The significance of the same requirements can be explained from a functional perspective. In order for a developed framework to be functional – that is, useful and practical enough for systems engineering – it is required that the framework possess the following –illities in the context of MDO: applicability, usability, flexibility, and extensibility. Applicability is a required trait so that the framework can scale well to large-dimensional problems with hundreds of design variables; be amenable to the application of various optimization algorithms; allow the monitoring of design progression; be compatible with heterogeneous, distributed computing environments; provide a centralized DBMS. Usability is closely related with GUI, which should be made intuitive, and visual programming interface for MDO problem formulation. Functional features that allow the real-time deletion, addition, and modification of process elements; debugging of the design process in a distributed computing environment; execution in batch modes; and cheap reanalysis from the last known point of analysis failure also contribute to the framework's usability. Finally, both flexibility and extensibility can be built into the framework if the requirements for object-oriented programming, integration of legacy and proprietary codes, support for new analysis codes as the need arises, and automation of any phase of the MDO process are satisfied.

2.4. Review of commercial MDO-enabling and academic MDO frameworks

At the time of this writing, a handful of software packages that are capable of serving as MDO frameworks exist in industry and academia. The two leading commercial software suites that represent the state-of-the-art in Process Integration and Design Optimization (PIDO) are ModelCenter, developed by Phoenix Integration Inc., and Engineous Software's iSIGHT family of products. Other notable players are MSC Software and PACE, a Germany-based company whose product lines encompass MDO-enabling Pacelab Engineering Workbench. In the United States (U.S.), past academic research efforts into MDO frameworks resulted in Intelligent Multidisciplinary Aircraft Generation Environment (IMAGE), which originated from Georgia Institute of Technology's Aerospace Systems Design Laboratory (ASDL), and NASA Langley Research Center's Framework for Inter-disciplinary Design and Optimization (FIDO).

ModelCenter is currently being touted as the preferred PIDO software for 7 of the top 10 aerospace companies and 9 of the top 10 U.S. defense contractors. If viewed as an MDO-enabling framework, the software consists of a front-end GUI and a back-end process integrator termed Analysis Server. From the GUI, a ModelCenter user can access his or her Analysis Server that exists locally or connect to multiple Analysis Servers over the Local Area Network (LAN) or Internet. The GUI portion of ModelCenter is currently exclusive to the Windows Operating System (OS), and supports a broad range of ready-made "Plug-ins," which allows intuitive wrapping of some of the most popular commercial CAD, CAE, and engineering programs such as ANSYS, CATIA, Excel, Mathcad, Matlab, STK, etc., to registered customers. Analysis Server supports both Windows and UNIX platforms (HP, SGI, Sun, LINUX, TRUE 64, etc.) and provides the function to wrap legacy codes or otherwise standalone disciplinary tools. The basic optimization package provides an optimizer that is accessible from the GUI. This optimizer is essentially a wrapped version of Vanderplaats R&D's Design Optimization Tools (DOT) and comes with 5 – 2 unconstrained, 3 constrained – optimization algorithms. The latest version of ModelCenter is compatible with CenterLink technology, which enables the creation of a Web-enabled grid computing environment from an intra-organizational network of heterogeneous machines.

Having its heritage in research conducted at General Electic (GE) Corporate R&D Center, the iSIGHT line of software products from Engineous Software Inc. has evolved into providing enterprise-level PIDO solutions to over 250 engineering companies. The "heritage" versions of iSIGHT, the latest of which is version 10, used to consist of farSIGHT, foreSIGHT, overSIGHT, and hindSIGHT. As an MDO-enabling framework, iSIGHT offers the means of customizing the simulation environment according to the system design problem at hand and applied MDO strategy with a unique scripting language called MDOL (which stands for MDO Language). The integration of most commonly used CAD, CAE, engineering analysis, optimization, and legacy tools is supported on both Windows and UNIX based platforms. Since 2005, Engineous has released a separate line of software titled iSIGHT-FD, which comes with a more intuitive GUI and leverages upon the company's FIPER technology. FIPER is a separate software package marketed by Engineous [18], and represents over 4 years of research conducted in partnership with GE, BFGoodrich, Parker Hannifin, Ohio Aerospace Institute, Ohio University, and Stanford University to realize a global-scale PIDO framework [19]. As a result, the GUI of iSIGHT-FD has a similar look, feel, and function to those of FIPER, having inherited its Java-based Object-Oriented structure.

Neither ModelCenter nor iSIGHT are advertised as MDO frameworks per se, but rather as PIDO frameworks. This is why it would be more accurate to describe them as MDO-enabling frameworks.

The genesis of IMAGE are documented in reference [20]. IMAGE is a GUI-driven and an internet-enabled design framework developed by Mark Hale, a past PhD student of ASDL. Originally conceived as an "open computing infrastructure that facilitates Integrated Product and Process Development from a Decision-Based perspective [20]". IMAGE is specific to the UNIX operation environment, having been developed using the Toolkit for Tool Command Language (Tk/tcl). The architecture of IMAGE consists of four major components, namely Developing Robust Engineering Analysis Models and Specifications (DREAMS), Available Assets, Agent Integration, and Computational Backplane. Among these four, Agent Integration is the key component that allows the creation of a customized MDO environment from Available Assets, if the intended usage of IMAGE is defined as the implementation of an MDO method in DREAMS. Lastly, Computational Backplane is the component that enables the analysis or design tasks to be performed in a distributed, heterogeneous computing environment.

As yet another GUI-driven framework, FIDO is composed of architectural elements that are similar to those of IMAGE. The management and execution of the design process is controlled by the Master component, while the Communication Library functions as both the Agent Integration and Computational Backplane of IMAGE. All available computational resources, such as analysis codes, in a distributed environment can be integrated into a cohesive whole by using the Communication Library, after being initialized by the Executive component. The Data Manager is responsible for enabling access to the central database, if one exists, and the user is able to observe the progression of analysis data or results in real time via the Spy component.

3. Globus Toolkit Ver. 4

3.1. Web services

Web services are modularized software components that support interoperable machine to machine interaction over a network. It can also be viewed as the product of the on-going paradigm shift in distributed computing technology, which is characterized by the melding of the XML and Internet protocols. For example, Web services utilize the Hyper Text Transfer Protocol (HTTP), easily lending themselves to be compatible with existing Internet protocols such as proxy and firewall [21].

The specifications of Web services are neither unique nor static, and there currently exist multiple definitions of what constitutes Web services in the literature. In this paper, Web services are henceforth understood to encompass, at a minimum, Web Service Description Language (WSDL), Universal Description, Discovery, and Integration (UDDI), and Simple Object Access Protocol (SOAP) for realizing messages, service interfaces, service publishing and service discovery. Since both WSDL and SOAP are based on the XML, which has quickly become the standardized format for data representation on the Internet, they are independent of computing platforms, operating systems, and programming languages.

As previously mentioned, GT4 is a middleware for implementing grid computing. In a broader context, however, it can become a useful tool for building improved Web services; e.g., circumvention of stateful service, enhanced security, etc. The Toolkit's many built-in features that provide the means to conveniently build Web services as components have led the authors to develop our Web services-based MDO framework using GT4.

3.2. GT4 architecture

The Toolkit is essentially a set of libraries and programs that can help address some of the most common obstacles in building a grid computing system. For instance, it comes with a VO Management Service as well as other helpful services, such as Security Service, Execution Management Service, Information Service, and Data Management Service [22].

GT4 allows the integration of available, albeit distributed, resources via Web Services, and also provides the following components related to Security Service: Delegation, Community Authorization, Authentication Authorization, and Credential Management. Grid Resource Allocation and Management (GRAM) Service is the sole enabler of Execution Management. The constituents of Information Service are Index, Trigger, and WebMDS. For Data Management, the Toolkit is supplemented by GridFTP, Reliable File Transfer (RFT), Replica Location Service (RLS), and Data Access and Integration (OGSA-DAI) tools. Additionally, GT4 provides the runtimes for Java, C, and Python [23].

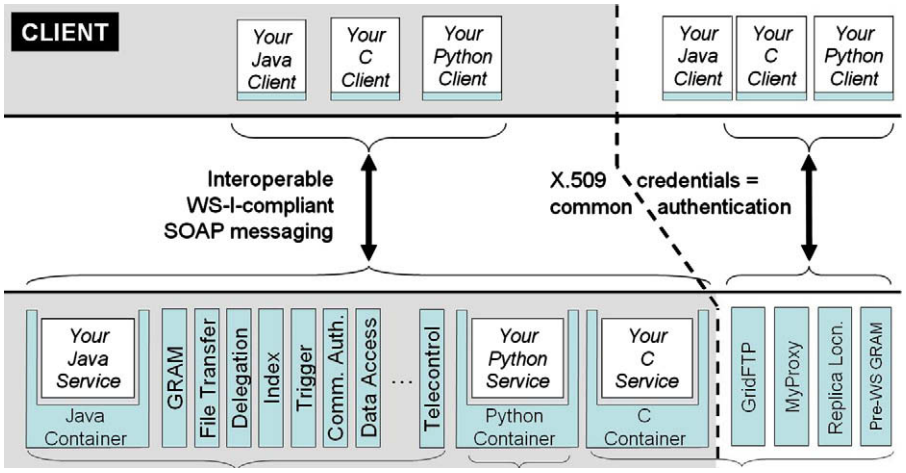

Fig. 1. GT4 Web Services Architecture.

Fig. 1 shows the GT4 Architecture. The shaded boxes in the figure represent the GT4-supplied codes, whereas the white boxes represent the user-built codes.

3.3. Building a web service with GT4

In order to build a Web services-based MDO framework using GT4, the first order of business is to develop all the disciplinary and cross-disciplinary resources that are to be accessible by the MDO framework as GT4 Web services. Web services are built on Web Services containers. Generally, a Web services container is a combination of a HTTP server, an Application server, and a SOAP Engine. A GT4 container consists of the standard Web services container described above, as well as the Toolkit libraries and handlers. While GT4 supports the development of services and client programs written in Java, C, or Python, all resources for our specific framework were developed as Java-based Web services.

Ensuring security across the computing environment, without costing the performance or scalability of the integrated resources, is a legitimate concern in a Web services-based MDO framework. It is preferable that no cumbersome security measures are imposed upon any of the design resources for optimum performance; one of the appeals of the framework such as ours is that the participating disciplinary experts are free to access the resources from anywhere and at any time. But then again, not all resources must be accessible to everyone. There may be situations in which certain tools and data cannot be shared due to proprietary reasons. Therefore, the formation of a VO that is well-balanced in terms of security and performance is an important issue that cannot be overlooked.

Fortunately, the GT4 supports the building of a VO through the Information and Security Services. The former provides Discovery and Monitoring services using Index and Trigger, supports VO creation through VO Index that collects information from every resource, and supplies WebMDS as a user interface [24]. The latter is accomplished by the Grid Security Infrastructure (GSI), which provides Transport-level and Message-level Security. This means that only those VO members that are cleared by through the GSI will have access to the MDO resources.

4. Web services-based MDO framework

4.1. MDO framework architecture

Our Web services-based MDO framework consists of Client, Management, Analysis, Optimization, CAD, and DB Services. The DBMS manages and controls the flow of both input and output data. Each service sends or receives messages through Web Services Interoperability (WS-I) compliant SOAP messaging. In the case where large amounts of data must be transmitted, GT4 services such as GridFTP and Reliable File Transfer (RFT) can be employed. Fig. 2 shows the architecture of our Web Services-based MDO framework.

The Management Service allows clients to efficiently utilize the available design resources. A VO Index has the index information of all services and is thus used to search and select the necessary Web services. A simple Certificate Authority (CA) routine issues certificates to the VO members so that each service can authenticate the users through those certificates [25].

The Client Service is the backbone of realizing a collaborative MDO environment. It can also monitor the status of the integrated resources through WebMDS. The practical operation of the framework from a user's perspective is described in detail in Section 4.4.

4.2. MDO agent

An MDO Agent conducts the individual execution of each design resource, and can be further classified into an Analysis Agent, an Optimization Agent, a CAD Agent, and a DB Agent. Each design resource is executed by its respective Agent.

Each design resource may be required to be handled differently under different computing platforms. As a result, wrapper and parser techniques, which transform SOAP messages into the formats understandable for each resource, become needed.

Each Agent first extracts the required data from a SOAP message via an appropriate parser, and then prepares the input files that are compatible with the data format of each resource using a wrapper. After the executions are finished, the Agent transforms the collected output files into SOAP messages. The trinity of the Agent, parser, and wrapper thus enable the automatic and organic execution of any MDO method.

The GT4 provides built-in services such as GridFTP and RFT service for Data Management [26] and Grid Resource Allocation and Management (GRAM) service for Execution Management [27]. These services support the rapid and reliable handling of large quantities of data, which is necessary in high performance computing. The instantiation of the Agent model within the GT4 framework thus provides an intelligent environment that is capable of efficiently using idle resources in solving even the largest of engineering design problems.

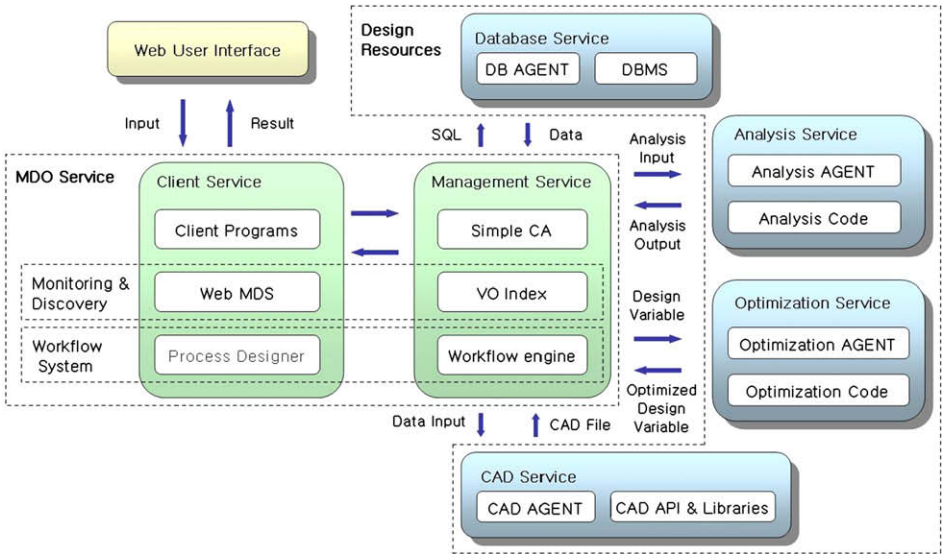

Fig. 2. Web Services-based MDO framework.

4.3. Workflow management system

A workflow is defined as the computerized facilitation or automation, in whole or part, of a process. The process to be automated is usually composed of multiple procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal. By extension, a Workflow Management System defines, manages, and executes workflows by the means of software, whose order of execution is driven by a computer representation of the workflow logic [28].

One can observe that MDO methods, in general, are inherently process-driven strategies composed of complex and interwoven execution steps. If the principles of the workflow management system can be adapted for implementation in an MDO environment, it is conceivable that automatic and organic execution of even the most complex of MDO methods could be implemented.

For the current research, a workflow management system that conforms to the five interfaces of the Workflow Reference Model given by the Workflow Management Coalition (WfMC) has been developed. Namely, the five interfaces are the Process Definition Interchange Interface, Workflow Client Application Interface, Invoked Application Interface, Workflow Interoperability Interface, and System Administration and Monitoring Interface. Fig. 3 shows the architecture of our Workflow Management System.

As it can be seen, the Workflow Management System consists of a Process Designer, a Workflow Engine, and a Workflow Client. The Process Designer consists of a Process Definition Tool, which is a visual-aid tool for defining a customized MDO strategy or process, and a BPEL4WS Translator, which converts the MDO method created using the Process Definition Tool into a Business Process Execution Language for Web Services (BPEL4WS) format. BPEL4WS is a Web Services standard technology and an XML-based Workflow Language for describing business process interactions, and hence the most suitable format for data storage in a Web Services-based MDO framework.

The Workflow Engine allows the execution of each design resource according to the pre-defined MDO process, and manages and controls all workflow paths. It is composed of a BPEL4WS Analyzer, a Process Handler, a Resource Classifier, a Worklist Handler, and a Process Monitor. The BPEL4WS Analyzer checks whether the Engine can execute the intended MDO strategy. The Process Handler executes the design resources as needed by the MDO process, and manages all steps from the beginning to the end. As its name implies, the Resource Classifier classifies the resources through the WSDL of each resource. The Worklist Handler is responsible for the communication between the Workflow Engine and the Workflow Client. The Process Monitor enables the Workflow Client to monitor the progress of the implemented MDO method. It sends the state information of each integrated resource as well as the state values of each design variable to the Workflow Client.

The Workflow Client is really a user interface. It sends the work lists processed by the Workflow engine to the human user, and subsequently informs the Workflow Engine whether or not the

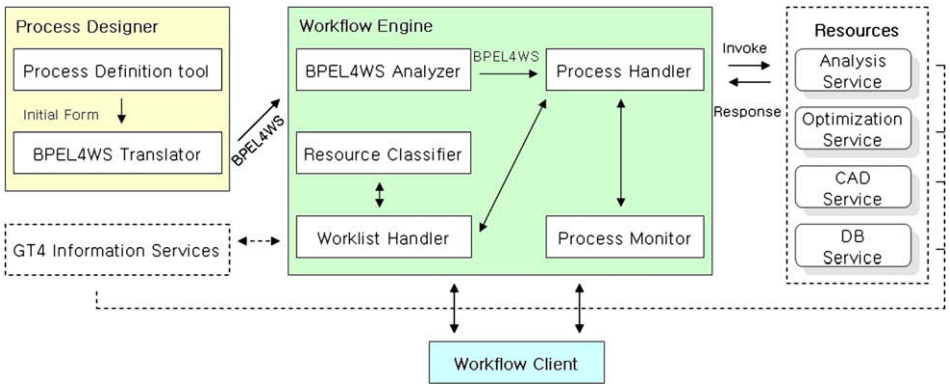

Fig. 3. Workflow Management System.

user has issued any commands. The Client also enables the user to confirm the progress state of both the MDO process and the design variable values.

4.4. GUI

By providing access to the Client Service through simple Web-browsers, the framework's GUI makes it possible for each participant of a multidisciplinary team to share a common MDO experience, regardless of his or her geographical locations. Any member of the team is able to review the analysis results after going through the following six-step, user interface process: Authentication, Project Generation, Task Generation, Definition of Design Variables and Constraints, Analysis and Optimization, and Confirmation of Results.

The Authentication step refers to obtaining a user account from the system administrator, and logging on to the framework to either create a new MDO environment or access an on-going work.

During the Project Generation step, a user is involved in inputting or creating the following information for administrative purposes: Project ID; Project name and date of creation; security access level; user ID and name; and Project description. A Project defines the scope of the attempted problem within the bounds of the overarching engineering design task at hand.

Similar to the previous step, Task Generation is where the user must enter or create the following information related to a specific Task: Task ID; Task name and date of creation; security access level; and Task description. Here, a Task refers to an MDO workflow, as discussed in Section 4.3.

This third step is also where a completely new Task can be defined or an existing Task can be retrieved from the database. A Task can be effortlessly defined via the Process Definition Tool and the Process Designer. The former allows the user to query the necessary design resources, while the latter enables the drag-and-drop creation of an MDO workflow from the identified resources. The workflow is subsequently saved to the database in the BPEL4WS format for future retrieval, modifications, or executions.

Once at least a single Task has been loaded on to the framework, then an optimization problem can be formulated in the fourth step: Definition of Design Variables and Constraints. Appropriately labeled Web-pages aid the user in the menial task of inputting the correct number of design variables and constraint functions.

The fifth step of the user interface process is Analysis and Optimization. During this step, the workflow engine executes the chosen Task as per the user inputs in steps three and four. The progress can be monitored in real time. Therefore, the user can terminate the process, adjust the parameter or Task settings, and restart this step as he or she sees fit.

Confirmation of Results is the final step in which the analysis and/or optimization results can be reviewed. The results can be numerical data pertaining to multidisciplinary analysis and/or optimization studies, or graphical data obtained from CAD Service, representing an optimum configuration to be opened by an appropriate CAD tool. All data files (results and inputs) can be saved to the database for future reference. Past data files can be just as easily retrieved.

4.5. The benefits of the framework

All modern commercial frameworks given in Section 2.4 facilitate the creation of an intra-organizational MDO environment. In terms of extensibility, however, the same existing frameworks do not easily lend themselves to forming a Virtual Organization, which represents an inter-organizational MDO environment, due to the fact that the frameworks were not coded using standardized computing technologies. Establishing a truly global-scale collaborative MDO environment, which is not restricted by the nature of the computing environment (homogeneous, heterogeneous, distributed, grid, etc.) or the geographical locations of the participating design/engineering/consulting organizations, thus appears to be challenging. Web services technology offers the means to assemble both disciplinary and cross-disciplinary resources according to the MDO needs of the VO. Web services technology is a standardized computing technology for pooling a wide variety of available resources. It is also highly extensible, making the technology a crucial architectural element of a software framework that aspires to be an enterprise-level MDO solution.

Agent technologies are adopted accordingly to enable seamless communication between the integrated resources, regardless of their geographical location and/or native operating system requirements. Each agent automatically converts the data format to one unified form that is determined to be suitable for a selected analysis code based on its knowledge, which enhances system's coherence.

One can observe that MDO methods, in general, are inherently process-driven strategies composed of complex and interwoven execution steps. The workflow management system invokes and executes web services in a predefined process and then one can avoid errors that might occur if these steps are manually processed. This fact indicates that the workflow management system can highly enhance system's reliability. The Process Designer is a visual-aid tool for defining a customized MDO strategy or process. The Workflow Engine allows the execution of each design resource according to the pre-defined MDO process, and manages and controls all workflow paths.

The Web services-based MDO framework that is proposed in this paper was intended to be highly extensible and flexible from its inception. It satisfies all of the requirements introduced in Section 2.3, and a powerful Web-based user interface was incorporated as part of the framework's standard features. Combined with the built-in grid computing capability, this allows the solicitation of expert disciplinary knowledge at anytime and from anywhere.

5. Conclusions

A novel Web services-based MDO framework, which enables the integration of various design-related resources through GT4, was developed. The framework offers an impressive capability to implement a desired MDO method in an organic and autonomous manner, largely due to our adoption of the Workflow Management System and MDO agent. If applied to a large-scale engineering problem, the built-in support for grid computing can potentially shorten the design phases in which cross-disciplinary analyses, trades, or optimization must take place. The framework was also made to be highly extensible by means of standardized Web services. Moreover, the framework can double as a collaborative MDO environment where all disciplinary experts can interact amongst themselves in providing guidance to future design activities. From the viewpoint of an MDO specialist or a systems engineer, such inputs can be seamlessly solicited via the Internet and Web pages, without being constrained by the inter- or intra- organizational structure. The knowledge gained from previous design activities would in turn lay the foundation for developing new domain-specific or generic MDO methods.

The flexibility of our MDO framework allows the implementation of any type of MDO strategy (via the Process Designer) at any phase of the product development cycle. Furthermore, our adoption of the Workflow Management System has enabled the framework to be applicable to a broad range of engineering systems design work. Being Web services-based, the framework is

especially capable of facilitating the creation of a VO. This alleviates the burden on the organization (or the department within an organization) spearheading the integration efforts to collect all required resources in one place. Regardless of whether or not the needed resources are distributed over an intra- or inter- organizational network, a user with the proper clearance can establish a secure connection to any disciplinary or cross-disciplinary resource, as long as it is accessible by the framework.

Future work with the developed framework will encompass its application to a variety of challenging aerospace design problems.

Acknowledgement

This paper was supported by Konkuk University in 2005.

References

[1] Current state of the art in multidisciplinary design optimization. An AIAA White Paper. Washington DC; September 1991.
[2] Sobieszczansk-Sobieski J. Sensitivity analysis and multidisciplinary optimization for aircraft design: recent advances and results. J Aircraft 1990;27(12):993–1001.
[3] Olds JR. The suitability of selected multidisciplinary design techniques to conceptual aerospace vehicle design. AIAA 1992-4791; September 1992.
[4] Frank PD, et al. A comparison of optimization and search methods for multidisciplinary design. AIAA 1992-4827; September 1992.
[5] Yoon KP, Hwang C-L. Multiple attribute decision making: an introduction, quantitative applications in the social sciences. Thousand Oaks, California: Sage Publications; 1995.
[6] Padmanabhan D. Reliability-based optimization for multidisciplinary system design. PhD Thesis, Notre Dame University; July 2003.
[7] Yun G, Ahn T. Case study of U-City project trend from a city planning perspective. SAMSUNG SDS Consulting Review. 2006;2:37–51.
[8] Wikipedia, Virtual Organization. <http://en.wikipedia.org/wiki/Virtual_organization>; June 2007 [accessed 12.07.07].
[9] Wikipedia, Open Grid Forum. <http://en.wikipedia.org/wiki/Open_Grid_Forum>; April 2006 [accessed 12.07.07].
[10] Sobieszczansk-Sobieski J. Multidisciplinary design optimization: an emerging new engineering descipline. In: Presented at the world congress on optimal design; 1993.
[11] Healy MJ, Kowalik JS, Ramsay JW. Airplane engine selection by optimization on surface fit approximations. J Aircraft 1975;12(7):593–9.
[12] Matheron G. Krigeage d'un panneau rectangulaire par sa périphérie. Note géostatistique, no. 28, CG, Ecole des Mines de Paris; 1960.
[13] Hajela P, Berke L. Neural networks in structural analysis and design: an overview. Int J Comput Syst Eng 1992;3(1–4):525–39.
[14] Braun R. Collaborative optimization: an architecture for large-scale distributed design. PhD Thesis, Stanford University; May 1996.
[15] Sobieszczansk-Sobieski J, et al. Bi-level integrated system synthesis (BLISS) for concurrent and distributed processing. AIAA 2002-5409; September 2002.
[16] Kroo I, Takai M. A quasi-procedural, knowledge-based system for aircraft design. AIAA 1988-4428; September 1988.
[17] Sobieszczansk-Sobieski J. Multidisciplinary design optimization MDO methods: their synergy with computer technology in the design process. Aeronautical J 1999.
[18] Wujek B, Koch P, Chiang W-S. A workflow paradigm for flexible design process configuration in FIPER. AIAA 2000-4868; September 2000.
[19] Roehl PJ, et al. A federated intelligent product environment. AIAA 2000-4902; September 2000.
[20] Hale MA. An open computing infrastructure that facilitates integrated product and process development from a decision-based perspective. PhD Thesis. Georgia Institute of Technology; July 1996.
[21] Header Kreger, IBM Software Group. Web Services Conceptual Architecture (WSCA 1.0). <http://www.ibm.com/software/solutions/webservices/pdf/WSCA.pdfMay2001>.
[22] Foster I. Globus Toolkit Version 4: software for service-oriented systems. J Comput Sci Technol 2006;21(4):513–20.
[23] Sotomayor B, Childers L. Globus Toolkit 4: programming java services. Morgan Kaufmann; 2005.
[24] Schopf JM, Raicu I, Pearlman L, et al. Monitoring and discovery in a Web services framework: functionality and performance of Globus Toolkit MDS4. Technical Report, Mathematics and Computer Science Division, Argonne National Laboratory; 2006.
[25] Welch V. Globus Toolkit Version 4 grid security infrastructure: a standards perspective. <http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf>; 2004.
[26] Allcock W, Chervenak A, Foster I, et al. Data grid tools: enabling science on big distributed data. In: SciDAC Conference, Institute of Physics Conference Series, vol. 16; 2005. p. 571–5.
[27] Czajkowski K, Foster I, Kesselman C. Agreement-based resource management. Proc IEEE 2005;93(3):631–43.
[28] David Hollingsworth. Workflow Management Coalition. The Workflow Reference Model. WFMC; 2001.