THE DESIGN OF A DISTRIBUTED, OBJECT-ORIENTED, COMPONENT-BASED FRAMEWORK IN MULTIDISCIPLINARY DESIGN OPTIMIZATION

By

Babak Mahdavi

School of Computer Science
McGill University, Montreal

August, 2002

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirements of the degree of Master of Science.

Your file Votre référence
ISBN: 0-612-88254-3
Our file Notre référence
ISBN: 0-612-88254-3

Abstract

The Multidisciplinary Design Optimization (MDO) can be defined as a methodology for the design of complex engineering systems where collaboration and abilities to mutually interacting between different disciplines are fundamental. The goal of MDO is to meet the needs for increased interdisciplinary interaction and communication and to reduce design cycle time. It provides a mean to integrate two or more design disciplines under the same optimization environment. It is a collection of tools and methods that permit the interaction between different disciplines involved in the design process. It enables the efficiency of designs to be optimized while supporting trade-off studies between the design objectives of diverse disciplines. Multidisciplinary analysis and design are often carried out by geographically dispersed engineering groups in a heterogeneous computer environment. In such a collaborative design environment, engineers working at geographically distributed locations can make a good design decision in a reduced timeframe, realizing lower product cost. In aerospace industry particularly, multidisciplinary optimization methodologies have necessitated the development of frameworks or problem solving environments capable to meet the needs of MDO practices. This framework can be defined as a hardware and software architecture that enables integration, execution, and communication among diverse disciplinary processes. In this thesis, Virtual Aircraft Design and Optimization fRamework (VADOR), a distributed, object-oriented, component-based framework enabling MDO practice at Bombardier Aerospace is introduced. The purpose of the VADOR framework is to enable the seamless integration of commercial and in-house analysis applications in a heterogeneous, distributed computing environment, and allow the management and sharing of the data. The VADOR distributed environment offers visibility to the process, permitting the teams to monitor progress or track changes in design projects and problems. Documentation of the MDO process is vital to ensure clear communication of the process within the team defining it and in the broader design team interacting with it. VADOR is implemented in Java, providing an object-oriented, platform-independent framework. The concepts of design pattern and component-based approach are used along with multi-tiered distributed design to deliver highly modular and flexible architecture.

Résumé

L'optimisation multidisciplinaire (MDO) peut être définie comme une méthodologie pour le design de systèmes complexes d'ingénierie de conception où la collaboration et les capacités à interagir mutuellement entre différentes disciplines sont fondamentales. Le but de la MDO est de satisfaire les besoins accrus d'interaction et de communication interdisciplinaires et de réduire la durée du cycle de conception. Elle fournit un moyen d'intégrer deux ou plus des disciplines de conception sous le même environnement d'optimisation. C'est une collection d'outils et de méthodes qui permettent l'interaction entre différentes disciplines impliquées dans le processus de conception. Elle permet d'optimiser le rendement des designs tout en supportant les études d'harmonisation entre les objectifs de conception de diverses disciplines. L'analyse et la conception multidisciplinaires sont souvent effectuées par des groupes d'ingénierie géographiquement dispersés dans une configuration matérielle hétérogène. Dans un tel environnement collaboratif de conception, les ingénieurs travaillant dans des endroits géographiquement distribués peuvent prendre des bonnes décisions de conception dans un espace de temps réduit, réduisant ainsi le coût du produit. Dans l'industrie aérospatiale en particulier, les méthodologies multidisciplinaires d'optimisation ont rendu nécessaire le développement d'infrastructures ou d'environnements de résolution des problèmes capables de satisfaire les besoins des pratiques en matière de MDO. Cette infrastructure peut être définie comme une architecture matérielle et logicielle qui permet l'intégration, l'exécution, et la communication parmi des processus disciplinaire divers. Dans cette thèse, Virtual Aircraft Design and Optimization fRamework (VADOR), une infrastructure orientée-objet, distribuée et basée sur des composants permettant l'utilisation de la MDO au sein de Bombardier Aéronautique est présenté. Le but de VADOR est de permettre l'intégration transparente des applications d'analyse commerciales et internes dans un environnement de calcul distribué et hétérogène, et de faciliter la gestion et le partage des données. L'environnement distribué de VADOR donne une visibilité au processus, permettant à l'équipe de surveiller le progrès ou de repérer les changements dans les projets et les problèmes de conception. La documentation du processus de MDO est essentielle pour assurer la communication claire du processus au sein de l'équipe qui le définit et de l'équipe de conception plus large qui interagit avec elle. VADOR est implanté en Java, fournissant un cadre orienté-

objet et indépendant de plate-forme. Les notions de modèle de conception, l'approche basée sur les composants et un design multi-niveaux distribué sont également employés pour fournir une architecture fortement modulaire et flexible.

Acknowledgements

"If I have seen far, it is because I have stood on the shoulders of giants."

I like this statement from Sir Isaac Newton which I assume can describe well my very strong gratitude towards people who helped me in one way or another in this thesis research project. I would like to thank my research directors Professor Ozell, my external supervisor from École Polytechnique, for his inestimable commitment and his immense wise guidance throughout the course of preparing and analyzing the material presented in this thesis, and Professor Newborn from McGill for his immeasurable devotion to helping see this project through to its final completion, and his generous and judicious direction.

I would also like to thank Professor Trépanier and Professor Guibault for their key roles in the VADOR project, as well as all the former and current professionals, students, and trainees involved in its development, contributed in one way or another to the advancement of this framework: Yun Wang, Abdulsalam Alzzubi, Amadou Ndiaye, Djamel Bouhemhem, Bin Chen, Daojun Liu, Qun Zhou, Christophe Tribes, Sébastien Carton and Franck Anouan.

I would further like to thank David Leblond from Bombardier Aerospace, who generously provided me with feedback on VADOR assessment in performing their selected processes at Bombardier. His presence at our weekly meetings, along with François Pépin, were of assistance in the constant evolution of the VADOR framework.

I would like to recognize Fassi Kafyeke from Bombardier Aerospace for supporting the VADOR project, J. A. Bombardier Foundation and the Natural Sciences and Engineering Research Council of Canada (NSERC), for their financial support, and Centre de Recherche en Calcul Appliqué (CERCA) for providing a pleasant R&D working environment.

Finally, I would like to thank Homayoun, Lise, Nagi and Shahrzad for encouraging me to continue and complete my master degree.

This thesis is dedicated to the soul of my father who departed to the heavenly kingdom during the first semester of my master programme at McGill.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

The increasing complexity of products in today's marketplace requires an increased need for more robust, accurate and less costly design processes. Today's advanced products challenge designers to attain performance requirements at affordable cost. The rapid pace at which new technologies and concepts enter these product designs severely limits the usefulness of past experience as a design guide [1]. Besides, advanced product design requires multidimensional analysis and simulation, usually within a very synergistic interacting coupled disciplines environment. Consequently, a new technology to aid designers working in such conditions throughout the design cycle is evolving; this technology demands an increasing reliance on processes that incorporate analysis and optimization. In the past two decades, vigorous research and development efforts by industry, universities, and government laboratories have established a foundation for the above process in mathematics, computer software and hardware, methodology, and engineering practice, which now coalesce into an emerging technology called Multidisciplinary Design Optimization (MDO)[1].

1.1 Motivation

Industry Canada [2] has identified MDO as one of 50 key technologies to be developed in the near future in order to maintain the competitiveness and

[1] Also for Multi-Disciplinary Optimization (MDO). In fact, different names have been used to describe this field of emerging technology. The terms such as: Computation-based Design, Simulation-based Design, Computational Prototyping, and the Concept of a Design Space are subtly referring to the same technology.

prosperity of the aerospace industry in Canada. The motivating factors for a framework or problem solving environment in the aerospace industry are the following:

- Aircraft design is multidisciplinary in nature
- Different disciplines execute independent of each other
- Potential exists for concurrent execution of some subtasks
- Hardware requirements vary with the discipline
- Large quantities of data and files are generated
- Potential exists for automating the design process

The aircraft industry has given considerable attention to MDO as manufacturers attempt to reduce the time-to-market of new products. Aircraft MDO practitioners would prefer to use high-fidelity analysis methods as early as possible in the design process, where the increased accuracy of the high fidelity methods can most strongly influence aircraft design. However, the existence of finite computational resources and time constraints limit the extent to which the high fidelity analyses may be applied in the early stages of the aircraft design process [3].

In the aerospace industry, the use of computational fluid dynamics (CFD) and computational structural dynamics (CSD) is now a part of the daily activities of engineers. Furthermore, analysis and design specialists also rely on additional "in house" programs tailored to the specific requirements of their tasks. The computational-based design environment then becomes highly complex and

specific to each application. Even within a single organization, each department has its own set of computational tools and programs, and which have little in common with those from other departments [4]. As a result of this complexity, the application of multi-disciplinary analysis and optimization (MDO) practices [5] has become desirable, although it faces a number of significant challenges, including collaboration, data sharing and management.

The challenge in applying an MDO methodology in a large aeronautical corporation lies mainly in the organizational aspects of engineering design. Mostly for historical reasons, the design departments are strongly segregated by disciplines such as "structures," "aerodynamics," "loads," "weights," and "stress," each department being responsible for specific aspects of the engineering work required to design an aircraft. As a result, challenges arise for several reasons [6]:

- Each department has its own vocabulary and methods and it requires a substantial effort to generate efficient multidisciplinary communication.
- Discipline specialists do not want to compromise on the tools used in their own discipline and recommend using state-of-the-art software. As a result, the computational resources required during a design optimization effort are enormous.
- Discipline specialists need to control everything in relation with the results produced by their codes, in order to ensure their proper usage.

- The transfer of information between departments is practically never automated and the transfer of data from the output of one discipline software to another discipline software frequently requires a several man-hours of processing by one or more persons.

As a result, many design or analysis problems are multidisciplinary; that is, they require the coordination of information from a number of highly specialized disciplines. Often the practice has been for specialists to independently optimize each discipline with limited direct interaction or communication with others. The development of computational frameworks offers the capability to take up these challenges via the use of sophisticated computational procedures combined with state-of-the-art optimization or design and analysis improvement techniques. The benefits of an integrated problem solving environment lie in the capability to support distributed analysis, to manage data flow between applications, and to access geometric design data and numerical drivers such as optimization methods.

1.2 Methodology and Objective

The term "methodology" is defined as a body of methods, procedures, working concepts, and postulates. Consistent with this definition, MDO can be described as a methodology for the design of systems where the interaction between several disciplines must be considered, and where the designer is free to significantly affect the system performance in more than one discipline [23]. In efforts to develop such a methodology that will be successful for aircraft MDO at Bombardier Aerospace and under the sponsorship of foundation J.A. Bombardier

and the Natural Sciences and Engineering Research Council of Canada (NSERC), Centre de Recherche en Calcul Appliqué (CERCA) is currently working on a project called Virtual Aircraft Design and Optimization fRamework (VADOR), to develop a framework for integrating multi-disciplinary design optimization, multi-fidelity engineering analyses programs and for managing the resulting analysis results. The main objectives of the VADOR project are:

- To develop a state-of-the-art software framework capable of supporting an MDO paradigm in a collaborative design environment.
- To implement within the framework, data management capabilities to closely follow the design data used and shared by the design team.
- To develop interfaces to selected in-house and commercial applications in use at Bombardier Aerospace.
- To deploy the framework at Bombardier Aerospace and train engineers in its programming and use.

All major players in aeronautics (Boeing[7], NASA[8][9], Airbus [10]) are currently performing research and development in a similar integrated design framework. As far as the technical engineering departments are concerned, the weakness of the integration precludes the application of the MDO methodology in the design cycle and a possible solution is to integrate the various analysis packages in a software framework. In all cases, specifically tailored systems are developed on top of various available distributed computing technologies.

The requirements for such MDO frameworks have been written in the form of specifications which will guide the software development. The specifications address four subjects: "Architectural Design," "Problem Formulation Construction," "Problem Execution," and "Access to Information." The "Architectural Design" specifications impose an object-oriented architecture, a natural GUI, the usage of standard languages and protocols, extensibility, and the support of collaborative design. The "Problem Formulation Construction" specifications require configurability through the GUI, the support of legacy applications and debugging facilities based on history traces. The "Problem Execution" specifications enforce that VADOR automates the execution of processes and the movement of data, support user interaction during the design process, allow users to operate in interactive mode or in batch mode and be fault-tolerant and flexible. The "Access to Information" specifications compel VADOR to provide database management features and provide tools to monitor the status of execution [11].

In fact, when the focus is on applications, one of the main objective of the framework, namely the management of the data, tends to be overlooked. When the focus is put on data, applications simply become methods required to produce or transform the data and the object-oriented paradigm reveals its fundamental nature. Accordingly, a good architecture for VADOR must be centered on the data. In the context of engineering design, data is usually stored in a file server and individual files can be quite large. In addition, these files are usually not self-describing and the appropriate management of this data requires

proper encapsulation into components with the required properties in order to meet the specifications for data management.

1.3 Scope of Present Work and Thesis Contribution

This thesis reports on VADOR, a new MDO framework developed to help engineers at Bombardier Aerospace to apply the MDO concept that enables the efficiency of design to be optimized and supports trade-off studies between the design objectives of diverse disciplines.

VADOR takes advantage of the Java to deliver highly modular, object-oriented portable design, using design pattern paradigm, as well as multi-tiered distributed architecture and component-based design approaches. This approach differs from other approaches in that all the existing frameworks fall short of satisfying the VADOR specifications, mostly because of their inability to store, in an appropriate fashion, the design data and the design decisions making a complete design project. In fact, most existing frameworks for integration and MDO propose mainly integration capabilities linked to optimization engines, and this addresses only partially the VADOR specifications.

The VADOR project was started in September 1999. By the time I joined the project (January 2000), the VADOR team consisted of three professors as well as a research associate. Currently, almost three years later, the VADOR team comprises ten people including three professors, two research associates and five students.

My main contributions to this thesis are:

Research

- A survey of the previous efforts made in this field
- A study and analysis of the technologies used in the advanced MDO framework structures, and identification of their strengths and weaknesses
- A feasibility study and an evaluation of the technologies and tools to be utilized

Requirements analysis and specification

- Identification and documentation of the exact requirements of the VADOR large-scale software system, particularly the *VADOR prototype*, *Executive Server*, *VADOR Search*, *Strategy builder* and *optimization*

Conceptualization and design

- Architectural and detailed design
- Specification of particular software systems that meet the requirements

Implementation and module validation

- Development of prototypes and codes
- Testing of individual modules

Integration and system testing

- Incorporation of all the individual modules and testing as a whole system

Delivery and maintenance

- Installation of the VADOR framework within the Bombardier designated infrastructure and modification of the system after the initial delivery as maintenance

These contributions aim to enable the large-scale VADOR framework to:

- capture in the framework the design methodology in use at Bombardier Aerospace
- collaborate with teams of engineers geographically distributed and simultaneously working on a design project
- automate the execution of analysis codes and data transfers
- implement single discipline and multi-discipline design loops
- track the history of a design project or piece of data
- identify the individuals working on a project, their respective responsibilities, and the status of their work
- keep track of the team designers' comments on a given design or result
- obtain a seamless transfer of data between applications
- illustrate the possibilities of the framework
- ensure a long-term benefit for the industrial partner

1.4 Overview and Structure of Thesis

Chapter 2 presents a background discussion and a general overview on Multidisciplinary Design Optimization and efforts on MDO framework developments. Chapter 3 covers the VADOR architecture and describes its various key components. The optimization issue and its implementation in VADOR are discussed in Chapter 4. Some case studies are presented in Chapter 5. Plans and recommendations for future work are explained in Chapter 6. Finally, Chapter 7 summarizes the research thesis work along with the conclusion.

Chapter 2: Background

Aircraft design demands a methodology that is more efficient, and sophisticated than the traditional serial design approach [12]. That is basically why it has been gradually replaced by Concurrent Engineering (CE) methodology, which is a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support [13][14][15][16][17][18][19]. The interaction of all participating engineering groups throughout a large-scale engineering design cycle is a multidisciplinary effort in a discipline such as aircraft design. In the 1980's, the effort to combine the concept of large-scale multidisciplinary engineering design and Concurrent Engineering led to the new invented research field of Multidisciplinary Design Optimization (MDO) [20].

2.1 Research in MDO's Conceptual Components

Some MDO research areas involve efforts in posing and solving large-scale engineering problems, the iterative system analysis, system simulation and MDO frameworks. The two main challenges of MDO are computational cost and organizational complexity. Ozell et al. [11], inspired by Sobieszczanski and Haftka [23], classify the MDO research fields into the following component groups: Approximation Concepts, Sensitivity Analysis, Decomposition Methodologies, Visualization, and MDO Frameworks.

The reduction of time and cost within the multidisciplinary design cycle is a very common goal of both the MDO and researchers in this area. Figure 1 shows

Sobieszczanski's principal components of MDO and their breakdown into more specific areas of research [21]. An MDO framework, for instance, can embody some or all of the found solutions in any of these seven principal areas, in different forms, such as module integration.

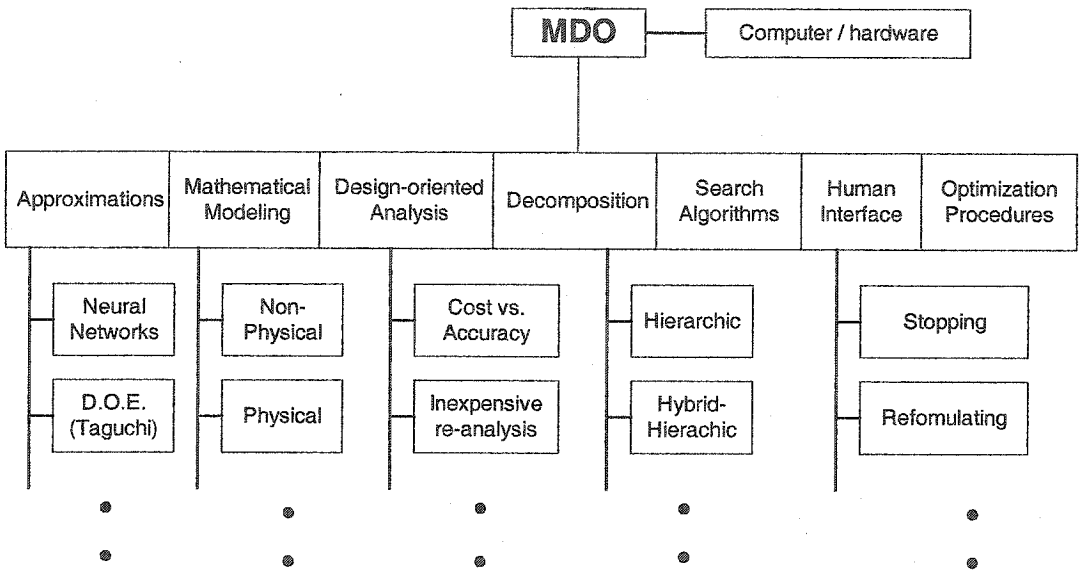


Figure 1: MDO principal conceptual components and their breakdowns

Ikoo [84] also gives a list of the research areas based on categories identified at National Science Foundation (NSF) workshop:

- Collaborative design tools and techniques
- Prescriptive models, design methods & normative theories
- System integration and infrastructure tools
- Design automation systems/tools
- Analysis, simulation, optimization tools
- Formal models of design process/design theories

- Design information access and support systems

2.2 MDO Technology Objectives

The fundamental objective of MDO technology is to develop an improved design capability while considering disciplinary interactions for synergistic affects.

Renaud [1] states that MDO technology is often comprised of (but is not necessarily limited to):

- analysis that is tailored to efficient repetitive use in design
- analysis that allows for trade-off of solution accuracy for computational cost
- sensitivity analysis at the discipline, component, and product levels
- optimization which similarly spans the range for detail to overall product performance
- accounting for uncertainties
- comprehensive, dynamic, and possibly distributed, large database management tools
- data visualization capabilities
- user interfaces that engage the designer in the process

2.3 MDO Framework

Rogers et al. [24] define a framework as a hardware and software architecture that enables integration, execution, and communication among diverse disciplinary processes.

2.3.1 Requirements for a Large-Scale MDO Capability

In the constantly evolving large-scale engineering design discipline, and particularly Multidisciplinary Design Optimization, Sobieszczanski [5][20][21][23] made an important contribution. In his review, Sobieszczanski identifies six important attributes that an MDO environment should have in order to support computational based design. These six attributes are[5][22]:

1- **Computer Speed:** the ability of the computer environment to provide answers rapidly, ideally in the order of seconds, in order to support the designer's creative train of thought.

2- **Computer Agility:** the ability of the software environment to provide a seamless transition between computational tools and models of various levels of accuracy, ranging from conceptual to detailed design.

3- **Task Decomposition**: the ability of the MDO system optimization software to allow teams of specialists to work concurrently on an MDO problem.

4- **Sensitivity Analysis:** the ability of the software environment to generate results on the sensitivity of the results to the variables controlled by the designer. This important information answers directly the *What if?*

question and provides the designer with knowledge without the need for re-analysis.

5- **Human Interface:** the ability of the MDO software and hardware environments to provide a designer with a form of data easily absorbable by human senses. This means the user interface to the MDO system should be through a natural and simple GUI and the analysis of results should use high performance virtual reality-type visualization for communication.

6- **Data Transmission:** the ability of the computer environment to transmit automatically, reliably, rapidly, and securely huge amounts of data.

These six attributes are for now only partially available to design engineers. Improvements are expected from various sources, including the increase in computer power (and particularly the efficient use of massively parallel computers), the increased reliance on computer science expertise and technologies, the improvement in efficiency of disciplinary optimizations and sensitivity computations, and the development of specific MDO methodologies and strategies. The design of a software framework should provide all these attributes for creating an ideal MDO environment.

2.3.2 Framework Objectives

The primary aim is to create a hardware and software architecture that provides support for multidisciplinary design optimization application development and execution. Most of the time, the resulting framework identifies means for reducing

the time and cost associated with the multidisciplinary design cycle. If MDO can be defined as: "How to decide what to change, and to what extent to change it, when everything influences everything else," [25] MDO framework objective can be described as such a framework software enabling these changes can be made. In general, the two most common objectives can be described as [26]:

- development of a state-of-the-art software framework capable of supporting the paradigm of MDO in a collaborative design environment.
- implementation in the framework of management capabilities to closely follow the design data used and shared by the design team.

2.3.3 Framework Requirements

Salas and Townsend [27] from the Multidisciplinary Design Optimization Branch (MODB) at NASA Langley Research Center (LaRC) propose framework requirements for an ideal MDO at LaRC's MDO Research. The requirements are summarized in the four following points of view:

1- **Architectural Design**

A framework should:

a) provide a Graphical User Interface (GUI) that is intuitive.

b) be designed using object-oriented principles.

c) be extensible and provide support for developing the interfaces required to integrate new processes into the system.

d) not impose an unreasonable amount of overhead on the optimization process.

e) be able to handle large problem.

f) support collaborative design.

g) be based on standards.

2- **Problem Formulation Construction**

A framework should:

a) allow the user to configure complex branching and iterative MDO problem formulations easily without low-level programming.

b) allow the user to easily reconfigure existing MDO problem formulations.

c) support the user in incorporating legacy codes (written in different languages) and proprietary codes (where the source is not available) into the MDO problem formulation.

d) allow the user to integrate discipline analyses with several optimization methods, including multilevel schemes involving sub-optimizations.

e) provide facilities for debugging of multiple processes on computers across a network.

3- **Problem Execution**

A framework should:

a) automate the execution of processes and the movement of data.

b) be able to execute multiple processes in parallel.

c) support execution distributed across a network of heterogeneous computers.

d) support user interaction (steering) during the design time cycle.

e) allow the user to operate in a batch mode.

4- **Access to information**

A framework should:

a) provide database management features.

b) provide the capability to visualize intermediate and final optimization and analysis results.

c) provide a monitoring capability for viewing the status of execution, including the system status.

d) provide some mechanism for fault tolerance.

2.3.4 Advanced MDO Frameworks

MDO frameworks and/or problem solving environments as a Research and Development (R&D) area includes universities, industries, and government research labs. Table 1 summarizes some studied developments in this active research area [11][27]. The following subsections briefly describe these products.

Table 1: Examples of advanced frameworks

Framework	Developer	Main Purpose
Access Manager	Boeing	Distributed software integramtion framework; Management of the computing process and data at Boeing ; Multidisciplinary environment that focus less on optimization but more on distributed heterogeneous computing
AML	TechnoSoft, Inc.	Modeling language for knowledge-based engineering; Focus on the data involved in the design; *Commercial product*
DAKOTA	Sandia National Laboratories	Multilevel parallel framework for design optimization
DARWIN	NASA Ames Research Center	Web-based framework to reduce design cycle by improving access to wind tunnel data; Focus on the data involved in the design
DeMAID	NASA LaRC	Design manager's aid for intelligent decomposition in multidisciplinary design
FACETS	University of New York at Buffalo	Simulation-based MDO framework
FIDO	NASA LaRC	MDO environment that focuses less on optimization but more on distributed heterogeneous computing; *DISCONTINUED*
IMAGE	Georgia Institute of Technology	Distributed computing and data management utilities; Focus on the data involved in the design; *DISCONTINUED*
iSIGHT	Engineous Software, Inc.	Optimization toolkit environment; *Commercial product*
LMS Optimus	LMS International	Virtual experiments and optimization based on simulation programs; simulation management and design space exploration; *Commercial product*
MDICE/ MDICE-AE	CFD Research Corporation	Multidisciplinary environment that focus less on optimization but more on distributed heterogeneous computing
MIDAS (OASIS, DEVO)	NASA Jet Propulsion Laboratory	Support integration for MDO in distributed, heterogeneous environment; Focus less on optimization; Other integrated optimization systems with MIDAS: OASIS, DEVO
NPSS	NASA Glenn Research Center	Enables multidisciplinary design and analysis of engines; Multidisciplinary environment that focuses less on optimization but more on distributed heterogeneous computing
Phoenix Integration	Phoenix Integration Inc.	Integrating multidisciplinary problems; *Commercial product*
Pointer / Epogy	Synaps, Inc.	Optimization and integration; Advanced computer aided exploration software for engineers; *Commercial product*
ProFES	Applied Research Associates, Inc.	Integrating MDO and probabilistic methods to perform RBMDO; Probabilistic FEA system
WICkED	University of New York at Buffalo	Web interface supporting parallel processing for MDO
XCAT/CCAT	Indiana University	Management of Computational Resources, High Performance Computing

2.3.4.1 Access Manager

Developed and deployed at Boeing Information and Support Service, Research and Technology, the Access Manager [28] software framework enables multidisciplinary design and optimization in a distributed heterogeneous computing environment. It supports a very flexible process control paradigm. Open architecture allowing the expansion of capabilities, object-oriented design, client/server architecture, coarse grained dataflow and parallelization and standard user-friendly Motif/X-Windows interface are some of its key attributes. Software is coded in C and C++, and the Remote Procedure Call (RPC) is used to enable distributed computing. Ridlon [28] states that Access Manager is designed to take into account identified needs of engineering users involved in MDO-type efforts such as the need to support existing applications without modification or access to the source code, the need to support iterative design and analysis and reuse of results, the need to support large quantities of data and large file sizes, as well as supports for long running, reusable processes and data and processes sharing in a distributed environment. With the deployment of the Access Manager by a number of different engineering organization in Boeing, Ridlon completes that the overall management of the computing process and the management of the data associated with that process is greatly facilitated using Access Manager framework. Unfortunately, there is no available information concerning the eventual current usage of the Access Manager in Boeing or its research and development discontinuation confirmation.

2.3.4.2 AML

The Adaptive Modeling Language (AML) [29][30] from TechnoSoft Inc, enables the multidisciplinary modeling and simulation of the product development cycle. The AML framework provides a web-enabled, distributed, collaborative design environment for concurrent engineering. Knowledge-based modeling is the basis for concurrent engineering and AML presents a unique object-oriented modeling paradigm to capture domain expertise into knowledge bases to assist in the design to production automation. AML is based on the LISP programming language. Scott [51] reports that efficient usage of AML requires familiarizing with the LISP language and good understanding of object-oriented programming practices in general. His report identifies AML's integrated parametric modeling capabilities as a strength and non user-friendy environment as one of the weakness for this framework. AML has been used at Lockheed-Martin in a design study.

2.3.4.3 DAKOTA

Design Analysis Kit for Optimization and Terascale Applications (DAKOTA) [31][32] is a noncommercial framework that provides optimization toolkit capability, developed by Sandia National Laboratories, a Lockheed-Martin company, for the US Department of Energy's National Nuclear Security Administration. Its object-oriented design using C++ provides a flexible, extensible, problem-solving environment as well as a platform for rapid prototyping of advanced methodologies that focus on increasing robustness and efficiency for computationally complex engineering problems. The same flexibility

and extensibility also permits the interface between analysis codes and iteration methods. This interface is intended to be very general, encompassing broad classes of numerical methods that have in common the need for repeated execution of simulation codes. The scope of iteration methods available in the DAKOTA system currently includes a variety of optimization, non-deterministic simulation, non-linear least squares, and parameter study methods. Eldred et al. [103] report that while DAKOTA was originally conceived as an interface between simulation codes and optimization algorithms, the new version expands to interface with other types of iterative analysis methods, such as uncertainty quantification with nondeterministic propagation methods, parameter estimation with nonlinear least squares methods, and sensitivity analysis with general-propose parameter study capabilities. It is a production tool for engineering design activities and a research tool for the development of new algorithms in optimization, uncertainty quantification, and related areas. DAKOTA can serve as a rapid prototyping tool for algorithm development. Data is exchanged between DAKOTA and the simulation code by reading and writing short data files. Access to the source code of the user's simulation software is not required. DAKOTA is executed through commands that the user applies in an input file. Distributed computing is supported using Message Passing Interface (MPI). DAKOTA has been used by Sandia to implement application on massively parallel machines [32].

2.3.4.4 DARWIN

The Developmental Aeronautics Revolutionizing Wind-tunnels with Intelligent systems for Nasa (DARWIN) [33][34] created at NASA Ames Research Center wants to redefine the classic approach to wind tunnel and other aerospace experimental testing. DARWIN is a framework for providing streamlined information access of experimental and numerical test facility data to aeronautical customers. The purpose of the improved information accessibility is to provide aeronautical engineers with essential elements to shorten and enhance the efficiency of wind tunnel testing in the design cycle process. Schreiner et al. [104] name five major product elements of the DARWIN information system as: Remote Access System, Integrated Instrumentation, Intelligent Database, Data Visualization and Infrastructure Enhancement. They are designed to work together to provide the aerospace customer of the future with the necessary access to information to greatly improve the design cycle process by gleaning more knowledge from available data and thus providing the capability to perform true design cycle iterations in a single test entry. DARWIN uses Web technology to access data.

2.3.4.5 DeMAID

DeMAID is not precisely a framework for optimization problem. Created by NASA, DeMAID stands for "Design Manager's Aid for Intelligent Decomposition." [79] introduces it in the following way: "The decomposition of a complex design system into subsystems requires the judgment of the design manager. DeMAID reorders and groups the modules based on the links (interactions) among the

modules, helping the design manager make decomposition decisions early in the design cycle." This corresponds to the primary goal of MDO, which is to decompose a large multidisciplinary system into a related grouping of smaller, more tractable, coupled subsystems. Rogers [80] from NASA LaRC, in his paper titled "Reducing Design Cycle Time and Cost through Process Resequencing" states that DeMAID minimizes the feedback couplings that create iterative subcycles, groups processes into iterative subcycles, and decomposes the subcycles into a hierarchical structure. The real benefits of producing the best design in the least time and at a minimum cost are obtained from sequencing the processes in subcycles. The DeMAID software contains a generic algorithm that rapidly examines many different sequences and selects the optimum sequence of processes within each iterative subcycle. It displays the processes in a design structure matrix format in which an element on the diagonal is any process that requires input and generates output. DeMAID is a knowledge-based software tool for reordering the sequence design processes and for identifying a possible multilevel structure for a design cycle [81]. It can be used to assist a project manager in making decisions that can potentially reduce time and cost of a design cycle [82].

2.3.4.6 FACETS

Framework for the Analysis of Coupled Engineering Techniques in Simulation (FACETS) [12] [83] is primarily interested in testing MDO methods and strategies on a simulation-based level by providing designers with an all encompassing computational infrastructure. By bringing together these numerous MDO tools

and techniques and making them available to a design manager, all in a single, all-encompassing infrastructure, such a tool can provide an MDO design manager with a powerful means for identifying possibilities for time and cost reduction within an existing multidisciplinary design. Developed in Multidisciplinary Optimization and Design Engineering Laboratory (MODEL) [58] of University of New York at Buffalo, FACETS contains a multitude of MDO tools and techniques intended for large-scale coupled system reduction. The ultimate purpose of FACETS is to provide a preliminary design tool that can enable a design manager to identify potential means for time and cost reduction within the elaborate multidisciplinary design process, in a simulation-based setting. FACETS also includes an optimization module, a system planning module, and an elaborate post-processor for result verification. This computational framework tool encompassing all of the research concepts provides an environment for simulating large-scale multidisciplinary design problems, and allows the user to explore numerous techniques and methods for solution. The benefit of FACETS is that it allows the user to quickly simulate the structure of a real-life coupled system, view its initial characteristics, perform some meaningful baseline calculations in simulation, and then view the final results. Thereafter, the user can then make judgements and subsequent modifications based on these results, and can quickly and easily re-run a new simulation in hopes of attaining better results and more useful insight to the true problem.

2.3.4.7 FIDO

The purpose of the Framework for Interdisciplinary Design Optimization (FIDO) [35][36] was to investigate the use of a distributed, heterogeneous computing system to facilitate communications, apply computer automation, and introduce parallel computing to produce a truly multidisciplinary process. Developed by Multidisciplinary Design Optimization Branch (MDOB) at NASA LaRC to demonstrate the technical feasibility and usefulness for selected distributed and parallel execution of a multidisciplinary analysis and optimization application by automating the coordination of analyses in various disciplines into an integrated optimization scheme, while allowing for visualization and steering by the designer. Although FIDO was not implemented as a generic framework for MDO applications, its development has provided much experience with the issues of framework architecture and problem formulation. Based on Parallel Vertical Machine (PVM) for distributed computing, FIDO had a *Communication Network* module for connecting computers, a *Communication Library* module to handle communications, a *Master Scheduler* for controlling the interactions between disciplines and a *Data Manager* for providing a central database connectivity. Lack of a research tool, platform-dependency, complex communications constructs, and complicated switching between discipline codes were some of the shortcomings of the FIDO framework reported by Sistla et al. [91]. The research and development on this framework have been discontinued at NASA LaRC.

2.3.4.8 IMAGE

Developed at Aerospace Systems Design Laboratory (ASDL) of Georgia Institute of Technology, the Intelligent Multidisciplinary Aircraft Generation Environment (IMAGE) was a research project of the Aerospace Engineering Department. Its modular and distributed computing architecture was used to assist in design activities such as posing design problems, assembling necessary analyses to tackle these problems, and executing them. Hale et al. [37] state IMAGE infrastructure is comparable to FIDO and other efforts in the development of underlying computing technologies. However, the IMAGE infrastructure is designed to explicitly support general design activities and an information model within an accountable design context. IMAGE was taking advantage of PVM for distributed computing and Berkeley Tk/tcl widget library for developing graphical user interfaces [38]. With IMAGE, a designer could build easily application-based scenarios to investigate a variety of engineering systems. El Aichaoui et al. [39] after doing several examples using the IMAGE system conclude: "Within the IMAGE environment, improved design can be achieved and high level of efficiency can be reached." IMAGE was based on object-oriented model with an emphasis on reusable components (agent-based architecture). The research and development on this framework have been discontinued at Georgia Institute of Technology ASDL.

2.3.4.9 iSIGHT

The original focus of iSIGHT [40][41] is on effective design optimization. This commercial software framework by Engineous Software, Inc., now available in

version 6.0, is a tool for integrating analysis codes and solving complex MDO problems. iSIGHT can considerably improve the efficiency of an MDO process by using various approximation models [42]. It provides an optimization toolkit that enables a combination of optimization methods such as Design Of Experiments[2] (DOE), Response Surface Modeling[3] (RSM), to be applied to the MDO application. Quality Engineering Methods (QEM) such as Six Sigma[4] are also implemented in the framework as well as Monte Carlo[5] simulation capability. Engineers can use GUI and its Multidisciplinary Design Optimization Language (MDOL) for constructing MDO problems. With MDOL, a user can create different type of 'blocks' which handles specific operations such as control flow of the design problem, simple internal calculation and system-level analysis methods. Integration of CATIA[6] a commercial finite analysis code used by most aerospace industry companies such as Bombardier, is also possible in iSIGHT. Sistla et al. [91] count the shortcomings of the iSIGHT in its handling of large problems, the requirement to learn MDOL and Tk/Tcl, the lack of database or file management and debugging capabilities, and its sequential processing on a single host computer. Scott [51] reports that iSIGHT's major limitations are its inability to

---

[2] Design Of Experiments (DOE) method allows to carefully control the number of simulations to run and to determine the search direction.

[3] Response Surface Modeling (RSM) is a mathematical model that approximates individual data points.

[4] Six Sigma is a highly disciplined process that focuses on developing and delivering near-perfect products and services. Sigma is a statistical term that measures how far a given process deviates from perfection. The central idea behind Six Sigma is that by measuring the number of "defects" in a process, one can systematically figure out how to eliminate them and get as close to "zero defects" as possible.

[5]Monte Carlo method is useful in assessing influence of variablity on the design after it is optimized.

[6] Computer Aided Three-dimensional Interactive Application (CATIA) is developed by DASSAULT SYSTEMES and marketed, distributed and supported by IBM. CATIA is an integrated suite of software applications covering all aspects of product design: Computer Aided Design (CAD), Computer Aided Engineering (CAE) and Computer Aided Manufacturing (CAM), whether by providing the necessary functionality to support collaborative product designs of all types, or the seamless integration that allows full support of company processes. http://www.catia.com/

easily distribute the execution of analyses to other machines, especially across platform boundaries. As for its greatest strengths, he mentions the system-level analysis and optimization because of a wide range of optimization methods including in this framework, and the ability to run these analyses in parallel in conjunction with its extensive system-level analysis capabilities. iSIGHT also includes a very detailed and flexible means for monitoring/reviewing the results. Boeing, Lockheed-Martin as well as Bombardier Aerospace are some example of aerospace industry users of this framework.

2.3.4.10 LMS Optimus

LMS Optimus made by LMS International [43][44] is an environment in which the user can automatically visualize and explore the design space to gain the critical insights into the dynamics of the problem. This commercial framework provides parallel processing capabilities, openness to multiple analysis codes, and an intelligent exploration of a range of algorithms to help a convergence to an optimal design. Nonlinear programming optimization techniques as well as DOE and RSM methods can be performed with this framework. The user employs the LMS Optimus GUI written in C++ and Motif to define analysis sequence. Inclusion of legacy code in the analysis sequence without modification is also possible. With the LMS Optimus, one can perform virtual experiments and optimization based on a sequence of multiple simulation programs. The tree representation of the database content, monitoring for failed analysis program runs, and interactive definition of the inputs, outputs and intermediate processes during a run using GUI are some features of the LMS Optimus.

2.3.4.11 MDICE / MDICE-AE

The Multi-DIsciplinary Computing Environment (MDICE) [45] developed by CFD Research Corporation provides an environment in which several engineering analysis programs run concurrently and cooperatively to perform a multi-disciplinary design, analysis, or optimization problem. Using MDICE, engineers are able to couple inherently dissimilar disciplines and programs from a variety of sources, performing distinct tasks such as geometry modeling, grid generation, CFD and structural analysis, and post processing into a single software system. This distributed object-oriented system can achieve a high degree of integration between the essential engineering analysis tools in a user-friendly environment. Automatic data transfer along with integration are among the strengths of this framework, while database capabilities is not. Paralleled distributed execution of codes is through PVM, and all codes can run on distributed computers.

There is also another project by CFD Research Center called Multi-DIsciplinary Computing Environment for AEroelasticity (MDICE-AE) [46]. The purpose of this project is to design a multi-disciplinary computing environment for performing dynamic aeroelastic calculations using almost any CFD and CSD analysis codes. The framework is based on a distributed object model.

2.3.4.12 MIDAS

The objective of the Multidisciplinary Integrated Design Assistant for Spacecraft (MIDAS) [47] project from NASA Jet Propulsion Laboratory (JPL) is to produce a graphic tool that allows designers with little or no computer experience to describe the design methodology that they use to design their parts of the

system, and to rapidly link together engineering and manufacturing tools to design the product. MIDAS supports integration for multidisciplinary analysis in a distributed, heterogeneous environment. As a design converges towards the design goal, the users can express the desire to visualize the progress via its interactive GUI. Graphical connection through interfacing the commercial products with MIDAS are also possible. There are two other optimization system development efforts at JPL that are integrated with MIDAS: DEVO and OASIS.

Design Evolver (DEVO) [105], a system for spacecraft design optimization, aims to provide an optimization tool that is seamlessly integrated into an existing computer-aided design (CAD) environment for spacecraft, which enables users to apply optimization algorithms, with a minimal amount of human effort. MIDAS is used as the design environment for DEVO.

Optimization Assistant (OASIS) [106][107], a self-configuring tool for automated spacecraft design optimization, improves system performance by automatically determining heuristic optimization strategies customized to specific spacecraft design optimization problem instances. The impact of using such a tool are the increasing quality of spacecraft designs generated, and the reduction of human expertise required to perform successful optimization. OASIS consists of an integrated suite of global optimization algorithms that are applicable to difficult black-box optimization problems, and an integrated intelligent agent that decides how to apply these algorithms to a particular problem. Given a particular spacecraft design optimization problem, OASIS performs a "meta-level" optimization in order to select an appropriate optimization technique to apply to

the problem and automatically adapts and customizes the technique to fit the problem. One of the major component of OASIS, called spacecraft design model, uses MIDAS as for its graphical design environment to allow users to integrate a system of possibly distributed design model components together using a graphical diagram representing data flow of the system.

2.3.4.13 NPSS

Numerical Propulsion System Simulation (NPSS) [48][49] is a concerted effort by NASA Glenn Research Center[7] to develop an advanced engineering environment for the analysis and design of aircraft engines and, eventually, space transportation components. Its purpose is to dramatically reduce the time (up to fifty percent), effort, and expense necessary to design and test jet engines. Using NPSS, engine designers will be able to analyze different parts of the engine simultaneously, perform different types of analysis simultaneously (e.g. aerodynamic and structural) and perform analysis faster, better and cheaper. An object-oriented approach is used to design and built the framework for NPSS.

2.3.4.14 Phoenix Integration

Phoenix Integration [52] solution includes Analysis Server and Model Center [53] developed by Phoenix Integration Inc. Product design requires the use of multiple applications often running on different platforms where data must be transferred between applications through the design time. The company's solution is to wrap

---

[7]The project is also supported by High Performance Computing and Communications Program (HPCCP) which aims to accelerate the development of high-performance computers and networks and the use of these resources in the Federal Government and throughout the American economy [50].

the applications, integrate the process, and then share the results. Analysis Server can wrap various software applications. Once wrapped, they can be securely published over the network (authentication is also possible [54]). Model Center allows access to these applications and exchange between them in order to get product optimization, a robust design and even a design for Six Sigma. Scott [51] compares Analysis Server operation to a Web server, which instead of hosting Web pages, hosts disciplinary analyses. Analogous to a Web browser, the Model Center program can access and utilize these wrapped analyses by negotiating with any number of Analysis Server programs being run on different machines. Phoenix Integration gets high marks for the intuitiveness and user friendliness of Model Center, as well as its flexibility in wrapping and distributing disciplinary analysis. The CATIA software can also be integrated into the system. Written in Java, Analysis Server can be used on various platforms from Unix-based to Macintosh. NASA, Boeing and Lockheed-Martin have already adopted Phoenix Integration in their aerospace design activities.

2.3.4.15 Pointer / Epogy

Before the Epogy release in September 2001, Pointer [55] name was used for both the optimization core and the process integration software. Both developed at Synaps Inc., Epogy combines Pointer now [56]. Pointer automatically controls a group of complementary optimization algorithms in such a way that they can efficiently solve a wide range of problems in a fully automatic manner. Pointer can be added to any existing parametric simulation software as an embedded application. In its combined form, Epogy is Synaps' Advanced Computer Aided

Exploration Software for Engineers. The software integrates the complex system of commercial and in-house simulation software tools required to perform multidisciplinary optimization tasks. Its Analysis Scheduler allows to graphically insert new analyses into the process. Conditional branching, parallel execution and sub-process execution are supported. Epogy allows the execution of in-house proprietary and commercial codes with no access to source. Epogy can be applied to any engineering discipline. For instance, Bombardier uses Epogy to improve wing design. Graphical stress representation such as NASTRAN[8] and other application in CFD, EDA, CAD can also be exploited. Airbus and Lockheed-Martin are among other customers.

2.3.4.16 ProFES

Sues and Cesare [85] present ProFES as a framework for integrating MDO and probabilistic methods to perform Reliability-Based MDO (RBMDO) [86]. RBMDO problems can be defined as a class of MDO problems wherein the system parameters (e.g., material properties, boundary conditions, loads, model prediction errors) are not necessarily deterministic and are described by probability distributions. For these problems, the objective is to maximize system performance (e.g., payload, aerodynamic efficiency) while satisfying constraints that ensure reliable operation. Since system parameters are not necessarily deterministic, the objective function and constraints must be stated

[8] NASTRAN, the NASA Structural Analysis System, provides a finite element analysis (FEA) capability for use in computer-aided engineering in aerospace research projects. NASTRAN is a standard in the structural analysis field, providing the engineer with a wide range of modeling and analysis capabilities.

probabilistically. For RBMDO problems the objective is to maximize expected system performance while satisfying constraints that ensure reliable operation.

The Probabilistic Finite Element System (ProFES) [87][88] application is developed by the Computational Mechanics Group of Applied Research Associates Inc. (ARA). ProFES enables the user to quickly develop probabilistic models of personal model executables and analytical formulations. It is a probabilistic Finite Element Analysis (FEA) system built on an innovative data-driven software architecture that seamlessly integrates state-of-the-art probabilistic mechanics techniques with commercial CAD software to make it practical and feasible to execute probabilistic analysis of complex structural components. The framework includes a GUI supporting a 3D graphical environment, a probabilistic analysis engine, public domain and commercial optimizers, and methods to rapidly integrate legacy and in-house applications as well as third party commercial applications for multi-disciplinary analysis. This add-on feature of ProFES permits work with commercial FEA such as NASTRAN. Only available on a Windows platform, creating customized external functions to extend ProFES' capabilities requires a 32-bit C, C++, or FORTRAN compiler with Windows application libraries. Industry partners in development of the ProFES development include: Pratt&Whitney and General Electric, both providers of aircraft engines.

2.3.4.17 WICkED

Becker and Bloebaum [57], in order to demonstrate the potential uses of Java for MDO problems and effectiveness of using the Internet as a communication tool,

present the Web Interface for Complex Engineering Design (WICkED), a software that simulates the convergence of a decomposed complex system in a distributed computing environment and computes the sensitivity derivatives of the system with respect to the independent input variables using Global Sensitivity Equation (GSE)[9] or finite difference method. Written entirely in Java, WICkED aims to support parallel processing for MDO problems on a network process. This is another development at MODEL [58] of University of New York at Buffalo.

This lab has other interesting ongoing research projects on MDO such as development of a framework for the solution of simulation-based coupled design problems in MDO, development of a virtual visualization environment for large-scale MDO problems, development of optimal convergence strategies for MDO, using Virtual Reality (VR) as an aid for solving design optimization problems, and approximation of GSE using RSM in MDO, most of them sponsored by NASA LaRC, Lockheed-Martin Tactical Aircraft Systems, and the National Science Foundation (NSF).

2.3.4.18 XCAT / CCAT

Both XCAT and its earlier implementation Common Component Architecture Toolkit (CCAT) [59] were developed at Extreme Lab of Indiana University. These CAT's (Component Architecture Toolkit) are software layers above a Grid[10] that

---

[9] One approach to MDO involves the use of the Global Sensitivity Equation (GSE) method. This method involves the computation of the total derivatives of a system by first solving for the partial derivatives of each subsystem. These partial derivatives are then used to obtain the total derivatives, algebraically.

[10] A grid [60] is a software framework providing layers of services to access and manage distributed hardware and software resources.NASA's Information Power Grid (IPG) [61] and Globus [62] are examples of grid.

enables users to make use of the Grid services in order to build and run distributed component-based applications. XCAT and CCAT are based on Globus Grid for its core security and remote task creation. However, a Grid alone cannot, in the near term, enable very large, single problems such as CFD calculations to be spread across distributed systems [63]. Both XCAT and CCAT are implementation of Common Component Architecture (CCA)[11]. The Extreme Lab introduces XCAT as an implementation of the CCA component specification that can be utilized to build Grid applications in two basic ways: via the generic application manager and control scripts, and via specialized components built for the application in question, either in Java or C++. Use of component technology, object-oriented design, and support of distributed heterogeneous resources through the use of Globus Grid make both XCAT and its earlier release CCAT very good and strong developments from a computer science point of view. However, the university-based nature of this product usually cannot interest industries because of the lack of support.

[11] The Common Component Architecture (CCA) specification describes the construction of portable software components that may be re-used in any CCA compliant runtime frameworks. CCA consist of two type of entities: Components and Frameworks. The philosophy of CCA is to precisely define the rules for constructing components and to specify the required behavior a component must exhibit and the interface between components and the framework. However, very little is said about the way the framework is constructed or the way the user interacts with the framework to connect components together. The reason for this is that there will be many different frameworks that can be used in very different situations. Some frameworks will be designed to optimize the use of components that are distributed across a wide-area grid. In other cases, the frameworks will be designed to optimize the composition of components that run on a single, massively parallel supercomputer. The goal is to provide a standard way that a component can be built so that it may be reused in any number of CCA compliant frameworks [70].

2.3.4.19 Review of the Advanced Frameworks Requirements

The framework requirements (see Section 2.3.3) review of presented advanced frameworks is presented in Table 3. It is important to mention that the purpose of this table is not a direct comparison of these frameworks with each others. In fact, the evaluation of these frameworks is mostly based on very limited available resources and information about them. Some were simply experimental while others are now discontinued or are basically not commercial products, which makes the accessibility to the resources even harder. It is obvious given, such conditions, that the evaluation is based on the limited and disproportionate information obtained from different resources regardless of the other frameworks estimation resources. The frameworks capabilities in regard to the requirements are indicated by the five qualifiers shown in Table 2.

Table 2: Qualifiers guide

meets or strongly meets the requirements	☺
meets or partially meets the requirements	😐
below the requirements	☹
not applicable	–
information missing	?

Table 3: Review of the advanced framework requirements

Framework	1a	1b	1c	1d	1e	1f	1g	2a	2b	2c	2d	2e	3a	3b	3c	3d	3e	4a	4b	4c	4d
Access Manager	☺	☺	☺	?	☺	☺	☺	?	😐	☺	?	😐	☺	☺	😐	😐	☺	☺	☺	☺	?
AML	😐	☺	☺	?	☺	☺	☺	?	?	☺	?	😐	☺	☺	☺	☺	😐	😐	😐	😐	?
DAKOTA	☹	☺	☺	?	😐	😐	☺	?	😐	😐	?	?	☺	☺	☺	?	☺	☹	😐	?	?
DARWIN	😐	?	?	?	?	😐	☺	?	?	?	?	?	😐	?	?	?	–	☺	?	?	?
DeMAID	😐	?	?	–	–	😐	😐	–	–	–	–	?	😐	?	?	?	–	?	?	?	?
FACETS	😐	?	?	?	?	😐	😐	?	😐	?	?	?	😐	?	?	?	?	?	?	?	?
FIDO	😐	☹	?	?	😐	☺	😐	?	☹	😐	?	?	😐	😐	😐	☺	?	😐	☺	☺	☹
IMAGE	☺	☺	😐	?	😐	😐	☺	😐	😐	☺	?	☹	☺	☺	?	😐	☹	😐	☺	☺	☹
iSIGHT	☺	😐	😐	?	☹	😐	☺	😐	😐	☺	?	☹	☺	☹	☹	😐	😐	☹	☺	☺	😐
LMS Optimus	😐	☺	😐	?	😐	😐	☺	☺	?	☺	?	😐	😐	☺	😐	☺	?	😐	😐	😐	?
MDICE	☺	☹	?	?	?	☹	☺	☹	😐	☺	?	☹	☺	☺	☺	☺	😐	☹	☺	☺	☹
MIDAS	😐	?	?	?	?	😐	?	?	?	?	?	?	?	?	😐	😐	?	?	?	?	?
NPSS	☺	☺	?	?	?	?	☺	?	?	?	?	?	?	☺	?	?	?	☺	?	?	?
Phoenix Integration	☺	☺	😐	?	?	☺	☺	☺	?	☺	?	😐	☺	☺	☺	☺	☺	😐	☺	☺	😐
Pointer / Epogy	☺	😐	😐	?	?	😐	☺	😐	?	☺	?	😐	☺	☺	😐	😐	☺	😐	😐	😐	😐
ProFES	😐	?	😐	?	?	?	?	?	?	☺	?	?	?	?	☹	?	?	?	?	?	?
WICkED	😐	☺	?	?	?	?	☺	?	?	?	?	?	?	?	☺	?	?	?	?	?	?
CCAT	😐	☺	☺	?	?	😐	☺	😐	?	☺	?	?	😐	☺	☺	😐	😐	☹	😐	?	☹

Chapter 3: VADOR Framework

This chapter focuses on an introduction to VADOR, its global framework architecture and various components.

3.1 Introduction to VADOR

The increasing complexity of engineering systems has sparked increasing interest in MDO, particularly in the aircraft industry. It is no surprise that Bombardier Aerospace, the main aeronautical company in Canada, and the third largest civil aircraft manufacturer in the world, makes efforts to get advantage of the so called "the concept of a design space."

Under the sponsorship of foundation J.A. Bombardier and the NSERC, CERCA has started working on Virtual Aircraft Design and Optimization framework (VADOR) since 1999. The first year was used to acquire an accurate definition of the framework specifications as well as an implementation of a prototype version. The first objective has been achieved by parallel investigations of the main aspects of the technical review and the selection of technologies for implementation, the selection and scheduling of interfacing tasks, and definition of the GUI. This included the study of Bombardier information management structures and a survey of various departments, including meeting with managers and engineers in order to get their feedback, analysis of the design cycle, and identification of all the applications that could be integrated and/or interfaced with VADOR [6]. In spite of the confidential nature of certain information concerning

Bombardier Aerospace and this project, the rest of this thesis tries to present and break down as much as possible the details while respecting this confidentiality.

3.1.1 VADOR Objectives

The main challenge in applying MDO technology in a large corporation such as Bombardier Aerospace lies in the organizational aspects of engineering design. Design departments being often segregated by disciplines, the transfer of information between these departments is partially automated. Most legacy applications being custom-made in-house, need specialized interfaces to communicate to each other. On the other hand, analysis process documentation and results data are essential for these organizations. VADOR aims to address all these issues by implementing the MDO methodology within Bombardier. The specific objectives are [26][64]:

- Development of a state-of-the-art software framework capable of supporting the paradigm of MDO in collaborative design environment.

  This objective permits:

  - capture in the framework of the design methodology in use at Bombardier Aerospace.
  - collaboration within a team of geographically distributed engineers working simultaneously on a design project.
  - automation of the execution of analysis and optimization codes and their corresponding data transfer.

- Implementation (within the framework) of management capabilities to closely follow the design data used and shared by the design team.

  This objective permits:

  - keeping the history of a design project or a piece of data.
  - identifying the people involved in a project and their respective responsibilities and work status.
  - keeping track of the various designers' comments on a given design and/or result.

- Development of interfaces to selected in-house and commercial applications in use at Bombardier Aerospace.

  This objective permits:

  - implementation of seamless data transfer between applications.

- Deployment of the framework at Bombardier Aerospace and train engineers in its programming and usage.

  This objective permits:

  - long term benefits for the industrial partner.

The main goal of this project is to provide technical engineers with an integration solution of legacy analysis applications and the associated data management tools necessary to handle the vast amounts of results data produced daily by the various departments of Bombardier Aerospace.

3.1.2 VADOR Specification

The specification of the VADOR framework [11][26] document meets the framework requirements described in Section 2.3.3.

The following are comments on these requirement points of view:

1. **Architectural Design**

   a) The interaction with the VADOR framework should be through a GUI which allows a graphical visualization of a design process, emphasizing on the flow of data within components. The GUI should also provide an advanced search tool.

   b) The object-oriented should be used in the design of the VADOR framework architecture. Java language is chosen for implementing the adopted object-oriented methodologies.

   c) The VADOR framework should be extensible. The user should be able to create new interfaces permitting integration of new process into the framework. Similarly to the programming language, using the basic key elements the user should be able to create extensible processes.

   d) The VADOR framework should aim to reduce the amount of overhead on an optimization process.

   e) The VADOR framework should aim to handle any MDO problems regardless of their size.

f) Multiple users should be allowed to access simultaneously. This implies that appropriate access permissions are defined and controlled to ensure the integrity of the data.

g) Basic standards of software components, languages and protocol should be used. Furthermore, this will ensure the longevity of the strategical and intensive VADOR framework usage in a real design environment.

2. Problem Formulation Construction

a) The configuration of complex branching and iterative problem formulation should be allowed. The GUI should enable configuration for a given problem to be as simple and intuitive as possible.

b) Reconfiguration of an existing MDO problem should be possible and simple.

c) The VADOR framework should enable the incorporation of legacy codes written in different languages and proprietary programs or applications for which there are no available sources. This would be possible by creating a proper generic wrapper mechanism and protocol in the system which permits these codes to work on their appropriate platform within the framework.

d) The VADOR framework should aim to allow the integration of user discipline analyses with several optimizations.

e) Facilities for debugging should be possible. Tracing an object history is one of the way to facilitate such a debugging. This requires that all objects should be able to report on their status. Log files can also be useful for debugging purposes.

3. Problem Execution

a) The VADOR framework should automate the execution of processes and creation of the data files. Processes are strategy objects that monitor the creation of the data files encapsulated in data objects.

b) Execution of multiple processes and their movement of data in parallel, should be supported.

c) In order to allow collaboration across the network, the VADOR framework should allow a transparent execution across a distributed network of heterogeneous computers while taking care of the corresponding data transfers.

d) Interaction during the analysis and design process should be supported. The user can halt the execution of a process, modify the data and resume it again or simply stop (kill) the process.

e) Users should be able to use the VADOR framework in interactive or batch mode. Once the project is launched, user can logout while the process continues to execute.

4. **Access to information**

   a) The VADOR framework should provide database management features in order to support a comprehensive data management capabilities.

   b) The intermediate and final visualization of the results should be possible whenever analysis codes permit it.

   c) Tools and/or widgets for monitoring the status of execution should be provided in VADOR GUI. The framework can inquire the status of any object at anytime and present them in appropriate manner to the user.

   d) The VADOR framework should be fault tolerant. A wrongly done operation should not result in neither the crashing of the servers, nor the jamming of the GUI.

3.1.3 VADOR Current Critical Review

Table 4 shows the VADOR complete critical review of the VADOR specification at the time of writing this thesis. A range of asterisks "*" from one to six is used to specify the level of the framework appreciation upon the specification criteria's points. In this way, one asterisk presents the lowest rank for absolutely not meeting the specification and six asterisks the highest rank for fully meeting the specification.

Table 4: Complete critical review of VADOR

Specification	VADOR
1a) Intuitive GUI	* * * *
1b) Object-oriented	* * * *
1c) Extensibility	* * * *
1d) No overhead on optimization	* * * *
1e) Handle large problem	* * * *
1f) Collaborative	* * * *
1g) Standards	* * * * *
2a) Configure complex branching	* * * *
2b) Reconfiguring existing problem	* * * *
2c) Incorporating legacy codes	* * * * *
2d) Integrating analyses with several optimization method	* * * *
2e) Debugging	* * *
3a) Automation	* * * * *
3b) Execute multiple process in parallel	* * * *
3c) Execution across heterogeneous computer	* * * * *
3d) User interaction	* *
3e) Batch	* * * *
4a) Database	* * * * *
4b) Intermediate visualization	* * *
4c) View status of execution	* * * *
4d) Fault tolerance	* * *

3.2 Object-Oriented Design by an Object-Oriented Tool

With respect to the VADOR specification (1b in Section 3.1.2), Java language has been chosen for the design and implementation of the adopted object-oriented methodologies in VADOR. Java is considered more than a language today. While Sun Microsystems introduces it as a *technology* [72], Berg and Fritzinger [73] demonstrate how Java language go from being a language to being an *industry*. Although Flanagan [78] insists on the importance of the distinction between the Java programming language, the Java Virtual Machine

(JVM), and the Java platform, deploying Java as a development tool makes it possible to take advantage of all its parts. Based on the power of networks, Java as whole enables the Internet and private networks to become a computing environment. A Java application can easily be delivered over the Internet, or any network, without operating system or hardware platform compatibility issues. This initially had an important influence on the language of choice since using Web technologies and tools such as Internet and a Web browser have been envisaged for the users of VADOR. In fact, the user of the VADOR prototype was also able to execute an MDO problem using a Web browser through a Java applet[12], although tight security restrictions within the same organization departments on one side and limitations of Java applets for both reading and writing files on the other side did not make it practical to take advantage of this Java feature for large-scale MDO problems.

Java package (and possibly subpackage) which permits to group related classes and define a namespace for the classes they contain, can also provide an ideal environment for the development of a large project such as VADOR. Two programmers located in two different offices, with poor communication, and responsible for different packages, can still name the classes without being concerned about the class name conflict. Although, Java now supports versioning by adding mechanism to discover and track changes of a piece of

[12] An applet is a little application, a small program that can be sent along with a Web page to a user. Java applets can perform interactive animations, immediate calculations, or other simple tasks without having to send a user request back to the server [74].

code, package, and JVM, other versioning tools such as CVS[13] can still be used to complete an ideal development environment.

Other considerations for choosing this object-oriented language among others are: its simple and complete development platform, portability or platform independence (cross platform), network-aware platform, central administration of new software versions, easy access to IT resources, rich and highly functional user interface, local data manipulation, technological unified network environment, simple and robust security model. Moreover, Java technology eliminates many of the problems associated with installing and running applications. Developing on the Java platform means that projects are completed faster and with less debugging. Becker and Bloebaum [57], in their paper entitled: "Distributed Computing for MDO Using Java" demonstrate that Java holds great promise for industrial MDO applications. Although Villacis [75] reports that the Java environment falls short for scientific computing, he concludes that Java is still useful as the glue for programming around scientific codes, while also reminding us of its rich object model, extensive set of core libraries, and simplified distributed computing framework.

3.3 VADOR Data Model

An effective framework enabling MDO practices provides the user with a flexible and configurable data model that can adequately satisfy the evolving requirements of engineers using computational-based analysis-and-design

[13] Concurrent Versions System (CVS) is a program that lets a code developer save and retrieve different development versions of source code. It also lets a team of developers share control of different versions of files in a common repository of files.

programs. Such a system must provide capabilities for the automation and integration of various processes used by engineers. It must also support and promote collaboration and data sharing. In the VADOR framework, collaboration and data-sharing are enabled through the use of components. Two basic kinds of these components, *DataComponent* and *StrategyComponent,* described in Sections 3.4.2 and 3.4.3, are respectively encapsulated design data and design methodologies. Their set of key attributes and types are stored in a database to provide basic data management capabilities in the system.

3.4 A Component-Based Development

Component-based development architectures have emerged as a standard design paradigm in many areas of application development. It is the building of software systems out of prepackaged generic elements, or, at least, from packages with well-known capabilities and precise functionality that can be encapsulated to create such a generic element [11]. The current trend toward this type of architecture is mainly the result of the convergence of the four phenomena originating from different perspectives:

1. **Scientific:** the progress of modern software engineering ideas with special emphasis on code reuse.
2. **Commercial:** the widespread success of theoretically unpretentious but practically useful techniques for building GUIs, databases, and other parts of applications out of single elements.

3. **Academic:** the generalization of object technology, which provides both the conceptual basis and the practical tools for building and using components.

4. **Market:** the push by some of the major players for competing technologies such as Common Object Request Broker Architecture (CORBA)[14], Component Object Model (COM)[15], Distributed Component Object Model (DCOM)[16], JavaBeans[17], and Enterprise JavaBeans (EJB)[18].

The goal of a component architecture is to simplify the application design process and speed application development [70]. An architecture of this kind, offering a component view of software, provides a way to implement more easily a network distributed framework.

---

[14] Common Object Request Broker Architecture (CORBA) [65] is an architecture and specification for creating, distributing, and managing distributed program objects in a network. It allows programs at different locations and developed by different vendors to communicate in a network through an "interface broker." CORBA was developed by a consortium of vendors through the Object Management Group (OMG), which currently includes over 500 member companies. Both International Organization for Standardization (ISO) and X/Open have sanctioned CORBA as the standard architecture for distributed objects (which are also known as components).

[15] Component Object Model (COM) [66] is Microsoft's framework for developing and supporting program component objects. It is aimed at providing similar capabilities to those defined in CORBA. COM provides the underlying services of interface negotiation, life cycle management (determining when an object can be removed from a system), licensing, and event services (putting one object into service as the result of an event that has happened to another object).

[16] Distributed Component Object Model (DCOM) [67] is a set of Microsoft concepts and program interfaces in which client program objects can request services from server program objects on other computers in a network. DCOM is based on the COM, which provides a set of interfaces allowing clients and servers to communicate within the same computer (that is running Windows 95 or a later version).

[17] JavaBeans [68] is an object-oriented programming interface from Sun Microsystems that lets you build re-useable applications or program building blocks called components that can be deployed in a network on any major operating system platform.

[18] Enterprise JavaBeans (EJB) [69] is an architecture for setting up program components, written in Java, that run in the server parts of a computer network that uses the client/server model. Enterprise JavaBeans is built on the JavaBeans technology for distributing program components to clients in a network. Enterprise JavaBeans offers enterprises the advantage of being able to control change at the server rather than having to update each individual computer with a client whenever a new program component is changed or added. EJB components have the advantage of being reusable in multiple applications.

3.4.1 Contract Aware Components

When integrating components into a larger framework, an important design task is to ensure that each component behaves exactly as expected. To this end, a component can be seen as a *service provider*, and the framework as an *interaction manager* for the movement of data between each agent. The goal of designing a proper environment for a framework then becomes the one in which the components are *contract aware,* i.e. there is a way of determining beforehand whether a component can be used within a certain context or not. Ideally, this information should take the form of a specification that describes what the component does without entering into the details of how. Furthermore, the specification should provide parameters against which the component can be verified and validated, thus providing a kind of contract between the component and its user. A proper classification of these contracts distinguish four levels [11]:

1. **Basic contract:** provides a simple component interface that lists all the operations and their signature (types of inputs and outputs) with no semantic properties. Static type checking verifies at compile time that all clients use the component interface properly, whereas dynamic type checking delays this verification until runtime. CORBA, DCOM, and JavaBeans all permit this type of contract to take place.

2. **Behavioral contract:** expresses what operations do, independently of how they do it. This permits developers to specify precisely every condition that can go wrong, and to assign explicitly the responsibility to

either the routine caller (the client) or the routine implementation (the contractor). Such contracts carry mutual obligations and benefits.

3. **Synchronization contract:** describes the sequence in which the operation will be executed. This becomes quite crucial in parallel programming where various operations must be synchronized and executed in the right order.

4. **Quality-of-service contract:** covers issues such as the maximum response delay, the average response time, the quality and precision of results, and the throughput of data streams.

Most of these contracts can be represented in form of diagrams by using Unified Modeling Language (UML)[19].

3.4.2 Data Component

As described in Section 3.3, there are two basic key components defined in the VADOR data model. DataComponent is one such component. It encapsulates the design-and-analysis data which is usually contained in a data file [76]. In addition to this simple or technical data usually stored in a data file, the requirements for the VADOR MDO framework forces it to keep a set of information tied to the data and encapsulated within the DataComponent. Examples of such attributes are presented in Table 5.

---

[19] Unified Modeling Language (UML) [71] is a standard notation for the modeling of real-world objects as a first step in developing an object-oriented design methodology. Its notation is derived from and unifies the notations of three object-oriented design and analysis methodologies: Booch's methodology, Rumbaugh's Object-Modeling Technique (OMT) , Jacobson's use case methodology . UML has been fostered and now is an accepted standard of the Object Management Group (OMG). Vendors of computer-aided software engineering products are now supporting UML and it has been endorsed by almost every maker of software development products, including IBM and Microsoft.

Table 5: Example of DataComponent attributes

Attribute	Basic Content
Owner	Object owner
Data file reference	URL of the encapsulated data file
Strategy ID	Identifier of the creator strategy
History	Parent(s), creation method and date
Access	Access permission
Validity	Level of confidence on data
Status	Data status
Comments	Any comments about object

There are two major kinds of DataComponent in the system, extending directly from it: *DCInstance* and *DCType*. DCIntance is an instance of a newly created DataComponent. A DataComponent has a type which serves as a mechanism for data standardization and validation. Furthermore, DataComponents can be classified by their respective types. DCType objects serve as type holders for DataComponents in the system. The size of a DataComponent is reasonably small considering the fact that a data file is typically stored in the file server(s) and only a reference to it is kept (in the form of a URL[20]). In this respect, many of the VADOR component attributes are so called *metadata* since they contain the description of data rather than the data itself. This is important since the instances of DataComponents are then stored in a database.

DataComponent can be *atomic* by encapsulating a data file (as described above), or *composite* by encapsulating one or more atomic or composite DataComponent(s). This user-defined hierarchical composition of

[20] Uniform Resource Locator (URL) is the address of a file (resource) accessible on the Internet. The type of resource depends on the Internet application protocol.

DataComponents enables the creation of a complex group of data files if necessary. Composite DataComponents are also classified by their types. A DataComponent requires a StrategyComponent (described in Section 3.4.3) to fill its encapsulated data file. An atomic example of a DataComponent could be a CFD_grid DataComponent, which encapsulates a mesh_CFD_analysis data file.

3.4.3 Strategy Component

DataComponent and StrategyComponent are both part of the VADOR initiative for making an abstraction of a complex engineering process. To achieve this level of abstraction, VADOR needed a way to encapsulate the programs and processes and then provide a mechanism to link them to the data. These executable programs, scripts, applications, or interactive graphical software applications usually require inputs in form of data files or interactive user entries and generally produce outputs (in form of data files). Execution of these legacy programs (in most cases) in a controlled sequence of operation normally representable in form of a flowchart corresponding to an algorithm, is what is defined here as a process.

The DataComponents described in Section 3.4.2 only encapsulate the technical data files used and shared by the design engineers. The StrategyComponents intend to capture the design process or methodology (strategy). This is done using *composite* StrategyComponents, which represent the basic methods and the data flow required to transform data in a given process. Compared to the DataComponent, it represents a higher level process. *Atomic* StrategyComponents are used to encapsulate executable programs,

scripts or applications that can create or fill the data files encapsulated in DataComponents. This can be seen as defining methods for creation of DataComponents. The required execution time for these programs can vary from a few milli-seconds to many days or weeks, depending on the engineering analysis to be performed. The type of StrategyComponent indicates the type of DataComponent that it can create. Sharing some similar attributes with DataComponent, Table 6 shows an example of StrategyComponent attributes.

Table 6: Example of StrategyComponent attributes

Attribute	Basic Content
Owner	Object owner
Type	Type of created DataComponent
Usage	Usage string of the encapsulated program
History	Parent(s), creation method and date
Access	Access permission
Comments	Any comments about object

This capturer of design process or methodology also enables the data flow standardization and documentation; that is, a better management of design strategies. Different kinds of StrategyComponent exist in order to construct a complex process, in the same way that a procedural language supporting parallelism (or at least pseudo-parallelism) provides basic mechanism permitting to describe a parallel algorithm. Instantiation, parallelism, testing, branching, and looping are possible in VADOR, using visually-presented and mouse-selectable different ranges of Strategies such as SequentialStrategy, ParallelStrategy, IfStrategy, WhileStrategy, ForStrategy, ConditionalStrategy. An example of a

complex StrategyComponent consists of the encapsulation of an optimizer, requiring use of OptimizerStrategy.

3.4.3.1 A Simple Procedure Example

The following steps should be performed for a simple process, shown as a flowchart in Figure 2:

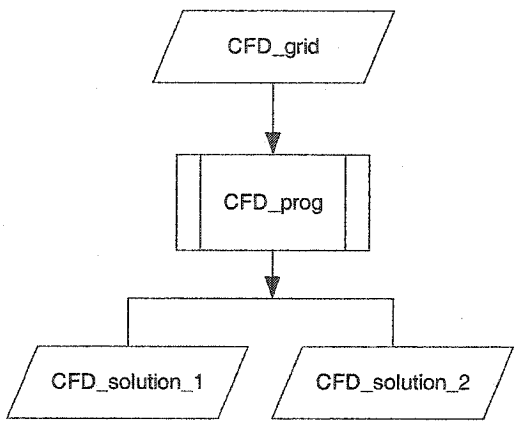


Figure 2: Example of a simple process

1. Definition of three DataComponent types encapsulating data files: CFD_grid, CFD_solution_1 and CFD_solution_2.
2. Definition of a composite DataComponent type encapsulating the data created by program CFD_prog, for instance, CFD_solution_1&2, containing respectively CFD_solution_1 and CFD_solution_2.
3. Definition of an atomic StrategyComponent, encapsulating the CFD_prog, which is capable of producing a composite DataComponent of type CFD_solution_1&2 and requiring an atomic DataComponent of type CFD_grid as input.

3.5 Designed by Patterns

The American Heritage Dictionary defines "pattern" as: "A model to be followed in making a thing." That thing in our context is an object-oriented software design. By observing a problem occurring repeatedly and by finding a solution for such a problem, one can create a model that can be used by everyone. To document such a pattern, one needs to describe the problem, describe the solution, illustrate the consequences of applying such a solution and finally, give a name to the pattern. Gamma et al. [89] define Design Pattern as "descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context."

A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems. It describes the problem, the solution, when to apply the solution, and its consequences. It also gives implementation hints and examples. The solution is a general arrangement of objects and classes that solve the problem. The solution is customized and implemented to solve the problem in a particular context.

Many of known and classified patterns have been used in the design of VADOR. Before starting to describe the major components in the VADOR architecture in the following section, a brief description of the actual patterns employed in the design of VADOR are given so they can later be used as reference to what (and why) they have been implemented.

1. Structural Patterns

   a) **Composite:** composes objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly. This pattern is used in design of composite DataComponent and composite StrategyComponent. They have a tree structure containing respectively other DataComponents or StrategyComponents.

   b) **Proxy:** (also known as **Surrogate**) provides a surrogate or placeholder for another object to control access to it. All the server components in VADOR provide proxies that permit a controlled connection to these servers from other components or servers.

2. Behavioral Patterns

   a) **Visitor:** represents an operation to be performed on the elements of an object structure. Visitor lets a new operation be defined without changing the classes of the elements on which it operates. For instance, in VADOR different types of ComponentStrategy such as ParallelStrategy and DoWhileStrategy can implement SgyVisitable interface in order to *accept* different visitors such as EexcuteSgyVisitor, SaveSgyVisitor and DeleteSgyVisitor, which implements ISgyVisitor interface, so they can perform different operations. Most of these interfaces are in *NetworkTools* package. A class diagram of this package can be found in Appendix B.

b) **Observer:** (also known as **Dependents** and **Publish-Subscribe**) defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. In fact, diverse VADOR components may generate several events. Listeners interested in these kinds of events can register themselves as observers, so they can be later notified if the specific event for which they have been registered happens. These observers can be VADOR users or other VADOR components in system. Most classes and interfaces enabling this mechanism are found in the VADOR Observer package.

c) **Mediator:** defines an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently. Some of the dialog boxes used in the VADOR GUI package encapsulate collective behavior in a separate mediator object which help to avoid a tedious individual customizing of every each of them by simply subclassing from a mediator class. For instance, the *VADORGUIDierctor* class in the VADOR GUI package is a mediator where other similar but different type of dialogs extends from it.

d) **Command:** (also known as **Action** and **Transaction**) encapsulates a request as an object, thereby letting parameterize clients with different requests, queue or log requests, and support undoable operations. Sometimes it is necessary to issue requests to objects without knowing

anything about the operation being requested or the receiver of the request. The *call()* method of *Vador_Agent* class (found in the *NetworkTools* package) can be invoked by other VADOR components such as VADOR Librarian without knowing anything about the operation implemented in the method.

3.6 VADOR Distributed Architecture

One of the requirements for VADOR, or any MDO framework in general, is to provide extensibility and support for new integrations into the system. The architecture for such a complex engineering design and analysis must, therefore, be flexible and modifiable. The goal is then to create an agile software framework.

Three-Tiered Architecture

Creating such an agile system demands a new architectural design rather than Monoliths and Two-tiered client/server approach. On the one hand, need for the custom interconnection between pieces of a system in the monolithic world which contains all the code needed to manage the data, implements the rules of application, and provides the user interface in a single mass, and on the other hand, difficult distributed system management in two-tiered systems leave the adoption of a three-tiered architecture approach as the best choice. The three-tiered architecture enables agile software in several ways. First, by treating software components as stand-alone *data providers*, *service providers* and *service consumers*, the three-tiered architecture creates a software infrastructure

of reusable parts. This reuse speeds the development and increase the overall system quality. Although, it is important to mention that there is more to component-based software reuse than just building the component themselves. Separation of all these component-based entities into different layers, make three-tired software applications easier to maintain and update. In the three-tiered architecture, applications are made up of cooperating collections of networked components. The switch from seeing components as integral parts of applications to seeing components as stand-alone entities which can provide services for applications, provides much of the power of the three-tiered architecture. Like any other good approaches, three-tiered architectures suffer from some drawbacks and limitations. Overhead of communication between the various layers causing some additional latency into the system is the more important one from the VADOR point of view. Figure 3, presents a global view of the VADOR architecture in a three-tiered appearance.

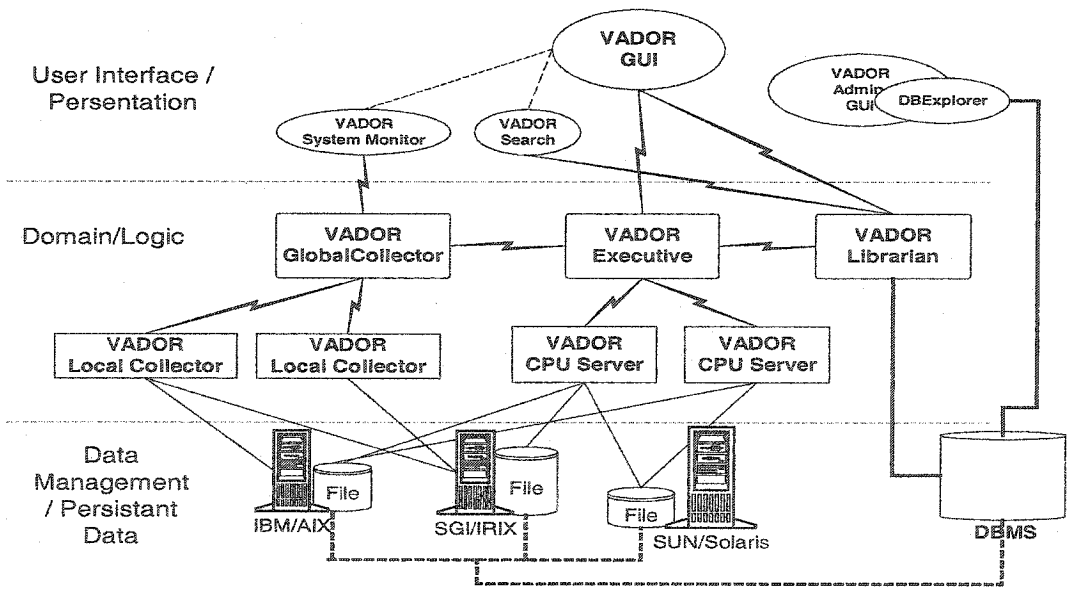


Figure 3: VADOR Architecture

The data provider component, identified as data management and persistent data tier in Figure 3, consists of the Database Management System (DBMS[21]), and actual persistent files on different machines. Service provider components in domain and logic tier include VADOR Executive, VADOR Librarian, VADOR CPU Servers, VADOR GlobalCollector, and VADOR LocalCollectors. Finally, service consumer represented in the user interface or presentation tier includes VADOR GUI, DBExplorer and all the small GUI applications that can be launched via VADOR GUI such as VADOR Search and System Monitor.

The three-tiered architecture seems like a minor extension to client/server approach which makes the difference a very subtle and critical issue, thus a pitfall to avoid. For instance, components of the data tier should provide service to the domain/logic tier or user interface tier without knowing which component made the request and from which tier. Generally, in a three-tired environment, it is not appropriate for components in the logic domain or presentation tiers to talk directly to data sources since this violates the encapsulation and abstraction maintained by each tier. Instead, component(s) in data tier should interface with the actual data storage mechanism and present an abstract interface to the other tiers while hiding the implementation details of how and where the data are stored. In the cases where it is not obvious to produce a canned query or interface for every possible query, components of the other tiers may directly

[21] A database management system (DBMS), is a program that lets one or more computer users create and access data in a database. The DBMS ensures the integrity and security of the data.

interface with database with the condition that data tier supports a SQL[22] mechanism for the other tiers. The data tier should still implement an abstract interface which minimizes the use of raw SQL and ease the future maintenance. This interface can eventually be developed in VADOR. Figure 4 shows a UML component diagram view of the global VADOR architecture.

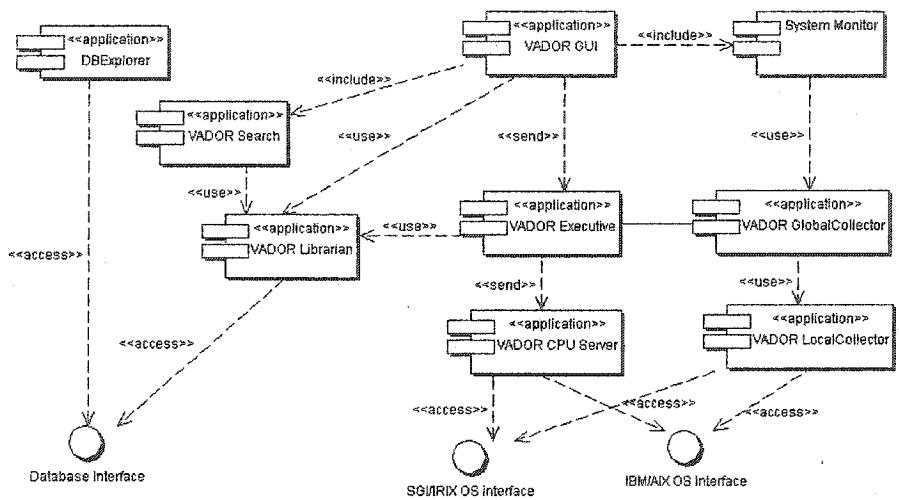


Figure 4: A component diagram view of the VADOR architecture

3.6.1 VADOR Librarian

A real librarian in a real library provides people with information and services related to the books, while at the same time he or she archives, files, manages and constantly updates information about these books in the database. VADOR Librarian is very similar to a real librarian with some differences. First, it is not a person but a Java server process, or a daemon. Second, instead of books it primarily manipulates the components, for the most part the instances of

---

[22] Structured Query Language (SQL) is a standard interactive and programming language for getting information from and updating a database.

DataComponents and StrategyComponents. Third, instead of people, it supplies services and information to the service consumer tier (VADOR GUI) or other service provider components such as Executive. VADOR Librarian provides a suitable interface to the VADOR database, permitting a persistent storage, retrieval and updating of information about component attributes. These attributes, as mentioned before, mainly keep the descriptive information or metadata so that only reference to a file (e.g. an URL string) is stored in the database and the actual file (potentially large) remains in the location where it has been created or moved to. Presently, the database used in the VADOR system is MySQL [98], a popular open source SQL database. VADOR Librarian uses Java Database Connectivity (JDBC)[23] API to connect to this relational database. Moving from MySQL database to any other relational database supporting a standard ANSI or ISO SQL requires only the loading in VADOR Librarian of the appropriate JDBC driver for the new database. If such a driver does not exist, it is still possible to use a bridge to connect JDBC to ODBC[24] using JDBC-ODBC bridge. In the case that the new database is not relational, a new interface should be written for VADOR Liberian in order to communicate with such a database.

---

[23] Java Database Connectivity (JDBC) is an Application Program Interface (API) specification for connecting programs written in Java to the data in popular databases. The API permits to encode access request statements in SQL that are then passed to the program that manages the database.

[24] Open Database Connectivity (ODBC) is an open API for accessing a database. By using ODBC statements in a program, it is possible to access to files in a number of different databases, including Access, dBase, DB2, Excel, and Text.

3.6.2 VADOR Executive

Execution of the diverse disciplinary processes is one of the fundamental tasks of any MDO framework. VADOR Executive is a Java server program which primary task consists of the execution of the design process captured within the StrategyComponents. VADOR Executive guarantees the automation of process execution, sequenced or paralleled, interactively or in a batch mode, within a distributed heterogeneous environment. Taking advantage of polymorphism combined with *Visitor and Composite* Patterns (see Section 3.5) in design of StrategyComponent, VADOR Executive can execute the different type of StrategyComponents simply by invoking a simple execute method. This ensures the automatized execution of the processes in a heterogeneous distributed environment. It is likely to have more than one Executive server within the VADOR system, every one of which is able to handle the execution of many different processes encapsulated in StrategyComponents. In fact, the Executive is a multi-threaded server, capable of managing multiple requests by the same user or different users simultaneously. The request of a service of each user, either an engineer or other VADOR server components, is kept track of as a thread with a separate identity. VADOR Executive makes sure that the status of execution on behalf of any thrown thread is kept track of until the execution is completed. A successful execution of a StrategyComponent (atomic or composite) ends with the creation of an atomic or composite DataComponent. A typical sequence of an execution request from VAOR GUI takes place in the following sequence:

- Receiving the execution request for a given process (identified by DataComponent instance ID that will be created as a result of this execution, passed as parameter) from VADOR GUI (running locally on the user machine).
- Launching a new thread for the new received request.
- Communicating with VADOR Librarian in order to recuperate the Data Component instance object and consequently the encapsulated StrategyComponent.
- Communicating with the VADOR CPU server(s) in order to delegate the actual execution of the StrategyComponent and their including StrategyComponent(s) in case of a composite strategy.
- Notifying VADOR Librarian at the end of the successful execution, permitting to add information about the new created instance(s) in the database and updating the status of the components.
- Sending a final message or an e-mail to the user after the final completion of execution. During the execution of a composite StrategyComponent, VADOR Executive also sends intermediate messages at the end of execution of any atomic StrategyComponent, which provides the user with information about the status of current execution and its progress.

Network Connection Type

Any interaction between these servers is impossible without a kind of communication connection. A network TCP[25] stream socket-based connection (found in java.net package) is used for the communication between VADOR Executive and other servers such as Librarian and CPUs. TCP provides a reliable session-based service and peer-to-peer (with each node having both server and client capabilities) mechanism. The stream provides the data transfer mechanism on top of the socket. With a stream socket, streams of bytes formed as packets are sent without errors and received in the same order that they are sent. Using this socket-based communication, component objects are serialized and passed from one server to the other. It is important to mention that this type of communication was not initially used in connection between VADOR Executive and the VADOR CPU server. Instead, a more sophisticated client/server mechanism using Java Remote Method Invocation (RMI) was implemented for this purpose. Java object-oriented RMI, considered a key component of the JavaBean architecture, is a lightweight mechanism which allows one Java application to call methods and access variables inside another application, which may be running in different Java environments or different systems, and to pass objects back and forth over this connection. Due to a callback problem in using Java RMI, which was causing an excessive and unreasonable additional time for VADOR Executive to receive any kind of notification from the VADOR

[25] Transmission Control Protocol (TCP) is a protocol used along with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet.

CPU server after the remote execution of StrategyComponents (thus a serious impact on the performance of VADOR), and as a result of unsuccessful investigation about the real cause of this problem, it has been decided to use a socket-based connection which is also used in communication between Executive and Librarian. CORBA [65] would be certainly a very good alternative (despite of its expensive licensing fee comparing to the free RMI) for all these communications. [90][91][92] provide more information concerning the potential usage of CORBA for solving MDO problems.

3.6.3 VADOR CPU

VADOR CPU is another Java server program designed to remotely execute the analysis task encapsulated in StrategyComponents. These servers are installed on a diverse range of computers with different architectures and operating systems, selected to be a part of the VADOR system. As explained in the previous section, VADOR Executive manages and coordinates task executions by delegating remotely the actual execution to the CPU servers. StrategyComponents restored in VADOR Executive, they encapsulate the design and analysis code, usually in a composite form; a tree structure containing other branches or leaves of composite or atomic Strategies. This tree structure is traversed by the Executive server and consequently, every actual atomic StrategyComponent leaf node, encapsulating a program, application or script, is sent (sequentially or in parallel, depending on its type) to these CPU servers for a remote execution. The CPU servers then run these legacy scripts or programs written in different languages such as C, C++ and Fortran able to run on a

specific platform for which they have been composed. This mechanism gives the impression that these legacy executable codes are *wrapped* in the way that they can be carried out within the VADOR framework regardless of the type of the programming language, or machine architecture and operating system on which they can run. That is why the CPU servers are also named and known as **Wrapper** servers in the VADOR terminology. Figure 5 shows how VADOR Executive receives the execution request from VADOR GUI, communicates with Librarian to recuperate that actual component, and then starts the execution procedure by dispatching the executable tasks to the VADOR CPU servers and consequently updates their status in the database through Librarian.

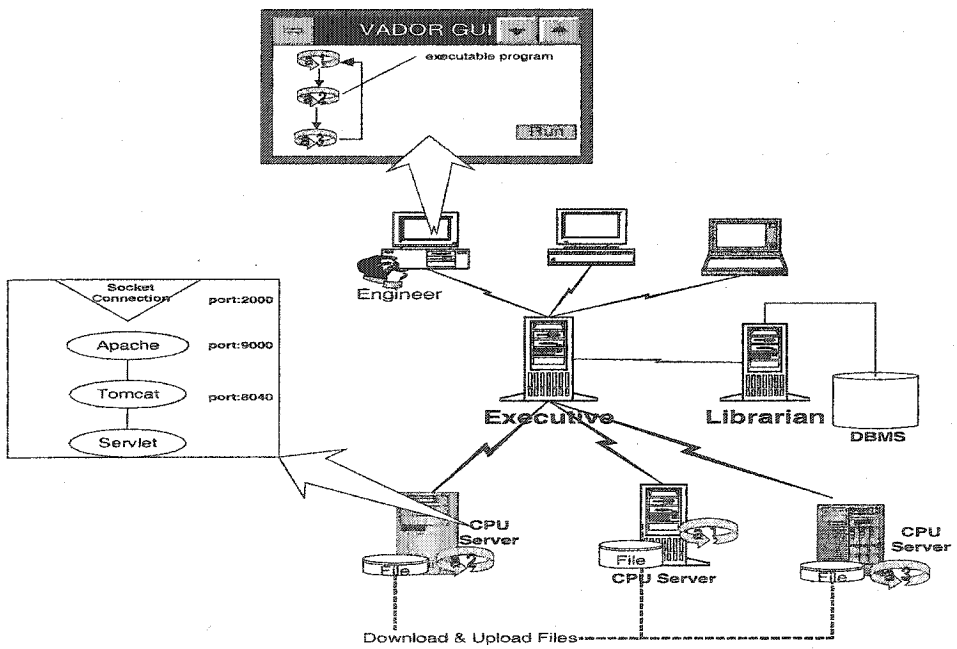


Figure 5: Process execution distribution to VADOR CPU servers

The choice of a CPU server by Executive for the execution of a task is not made at random. There are a range of different machines which are classified by their architecture and their operating system, for instance IBM/AIX, SGI/IRIX,

SUN/Solaris. VADOR administrator makes sure that a VADOR CPU server runs in every machine that is assigned to host the remote execution. The selection of the hosts is made by the engineer who creates the process. Therefore, the name of a target host is specified in the StrategyComponent, which indicates to Executive where the StrategyComponent should be executed. Usually, there may be more than one CPU server for a class of machine. In this case, the load balancing system (discussed in Section 3.6.4) in VADOR and its designated servers for this purpose can help users and Executive to monitor the system and send the execution where the load of task is lower.

To run a task, a CPU (Wrapper) server performs the following steps [93]:

1. Locates the executable program, application, or script file
2. Builds a temporary directory
3. Transfers all input files to the temporary directory
4. Runs the program, application, or script file
5. Transfers the created output files
6. Cleans up and destroys the temporary directory

Synchronization of all these steps is guaranteed by the CPU servers, which, similarly to the Executive server, use multi-threading for hosting other executions at the same time.

Generally, the input and output files are not kept locally on the machine where the execution takes place. In such a condition, the file transfer mechanism is necessary to download and upload all the necessary files. In order to download

files, the CPU server uses the Apache server (designed normally for downloading web-context files), however, for uploading output files, the CPU server need to use a Java servlet module which in turn requires a Tomcat server to be added to the Apache server in order to validate Java servlet's functions. With such a custom transfer protocol, based on standard and widely available servers, the CPU servers can perform a secure transfer of files, necessary for the remote execution of the legacy engineering analysis codes within the VADOR system.

3.6.4 Load Balancing

VADOR is not a run-time system, however, because it manages and executes different programs in parallel, the performance of the system is still considered a crucial issue. In order to improve the performance of such a distributed system, a load balance mechanism needs to be used to reach an even workload across the CPU servers. The goal to achieve for the VADOR Executive as a process manager, is to allocate the processes to the CPU servers, by making sure that workload is spread around the available CPU servers as to make full and balanced use of the computing power available. This goal aims to avoid a situation where some CPU server hosts are overloaded while the others are idle.

Load balancing is the equitable distribution of workload among the computing resources available in a network [95]. The primary function of load balancing is to recommend decisions that improve performance [96]. There are basically two kinds of load balancing algorithm: *static* and *dynamic*. Static load balancing algorithms are simple and have low overhead. They rely on the estimated

execution times of processes and inter process communication requirements. This may cause that some CPU servers be idle when others are overloaded, hence unsatisfactory for a parallel-able server. The "taking-turns" Round–Robin algorithm falls into this category. On the other hand, dynamic load balancing algorithms permit adaptation to changing circumstances. These algorithms which make run-time decisions based on system state can be defined by their implementation of the following policies [95]:

- **Information policy:** specifies what load information is to be collected, when it is to be collected, and from where. It is important that the load information takes into account not only the needs of a task, but also I/O and memory operation requirements.
- **Placement policy:** determines where a process should be located or transferred to.
- **Transfer policy:** describes the conditions under which a process should be transferred.

These algorithms should also take into account the nature of the tasks which can be: CPU-bounded, memory-bounded, or IO-bounded. For instance, a CPU-bounded task should be assigned to a host with the lowest CPU load. In the same way, a memory-bounded task cannot be assigned to a host with not enough memory resources available. These load balancing algorithms should be used in the way that the entire system loads will be estimated. This estimation may contain the information load consisting of: system CPU usage, CPU idle, current free memory, system buffer activities, and page fault.

In VADOR, the load balancing mechanism is enabled by implementing two different kinds of Java server programs, described in Sections 3.6.4.1 and 3.6.4.2.

3.6.4.1 Global Collector

This server is responsible for collecting CPU servers' host load information across the network by applying the load balancing algorithm. The GlobalCollector server periodically gathers system load information by communicating the LocalCollector servers and ensures the proper management of these information. The management of the LocalCollector servers is also under the responsibility of GlobalCollector. If one of LocalCollectors crashes, it will be restored by GlobalCollector. Obviously, VADOR Executive is the first client of GlobalCollector. However, the user can also utilize the *System Monitor* frame in VADOR GUI to visualize the load information on available hosts, via the VADOR Executive connection, and possibly assign an explicit execution of a StrategyComponent on a selected host. In the absence of GlobalCollector, the Executive server may continue to apply the random algorithm to assign an execution task to a selected CPU server among a range of available CPU servers for the same class of machine/OS architecture.

3.6.4.2 Local Collector

The LocalCollector Java servers are basically present in every host where a CPU server is running. They periodically collect information about a host, provide a load information estimation, and compare it with the previous collection. If a change is observed as the result of this comparison, they notify GlobalCollector

by establishing a communication channel, otherwise, they can sleep until the next pooling loop. GlobalCollector can also explicitly require them a load information.

3.6.5 VADOR GUI

The VADOR GUI is the main interface between users and the other part of the VADOR framework. It mainly interacts with the Executive and the Librarian servers. Efforts have been made to create a user-friendly environment where users can intuitively exploit the framework. With VADOR GUI, engineers can use different available builders to create DataComponents (atomic or composite), along with their types, and StrategyComponents (atomic or composite). A browser can be used to navigate through the system directory folder to view, open, or delete the instances of DataComponents and StrategyComponents. Different views of DataComponent and StrategyComponent are provided by choosing the appropriate icons in the GUI.

VADOR GUI is multi-tasking environment, means that more than one application can be used within the same GUI frame. For instance, the user can launch VADOR Search (described in Section 3.6.5.1), and VADOR Explorer at the same time while building an atomic Strategy via the atomic Strategy builder. A view example of VADOR GUI is presented in Figure 6. More images of other parts of VADOR GUI, widgets and tools, including VADOR Explorer and different component builders can be found in Appendix D.

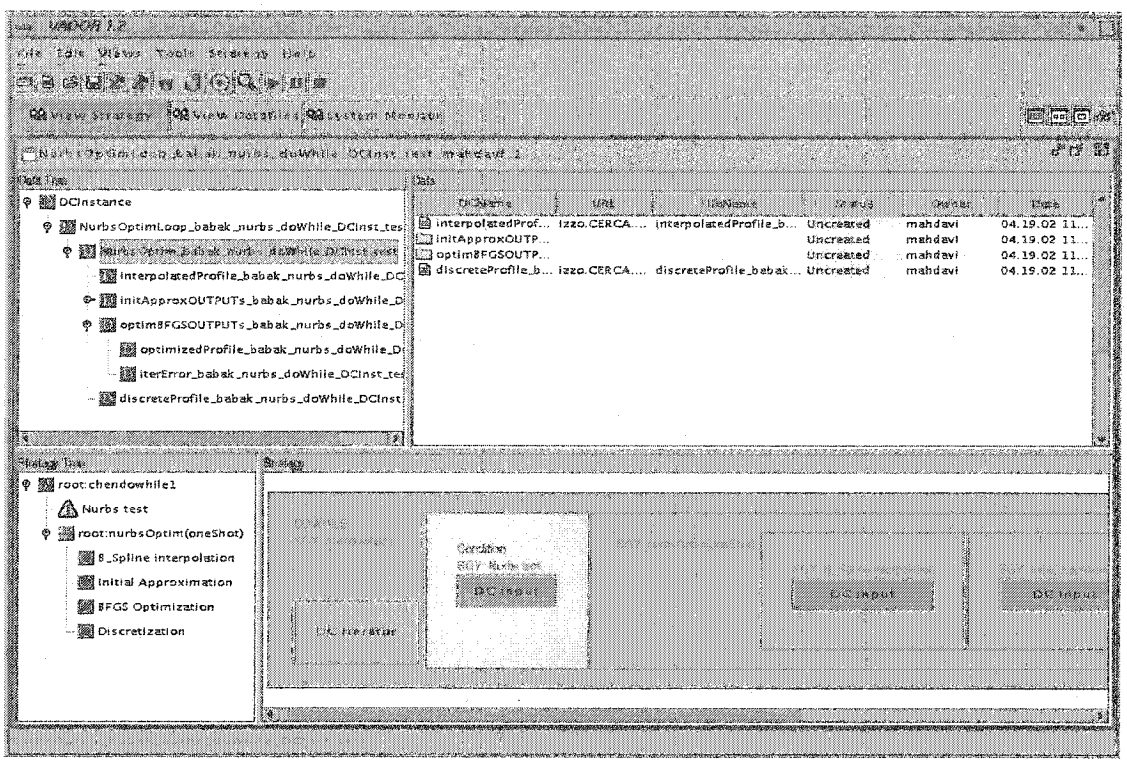


Figure 6: VADOR GUI

3.6.5.1 VADOR Search

With the VADOR Search tool integrated into VADOR GUI, users can perform a very user-friendly search on the different VADOR components attributes, stored in the database. The user does not need to have any knowledge of database language since SQL queries are built transparently. The results of the research are presented in the Java table (JTable) or in the form of a tree structure, if the search is recursive. Users can ultimately do different permitted operation on the found results, although this operation is still not completely implemented. VADOR Search utilizes Liberian in order to get the results out of the VADOR database. Figure 7 shows a view of VADOR Search.

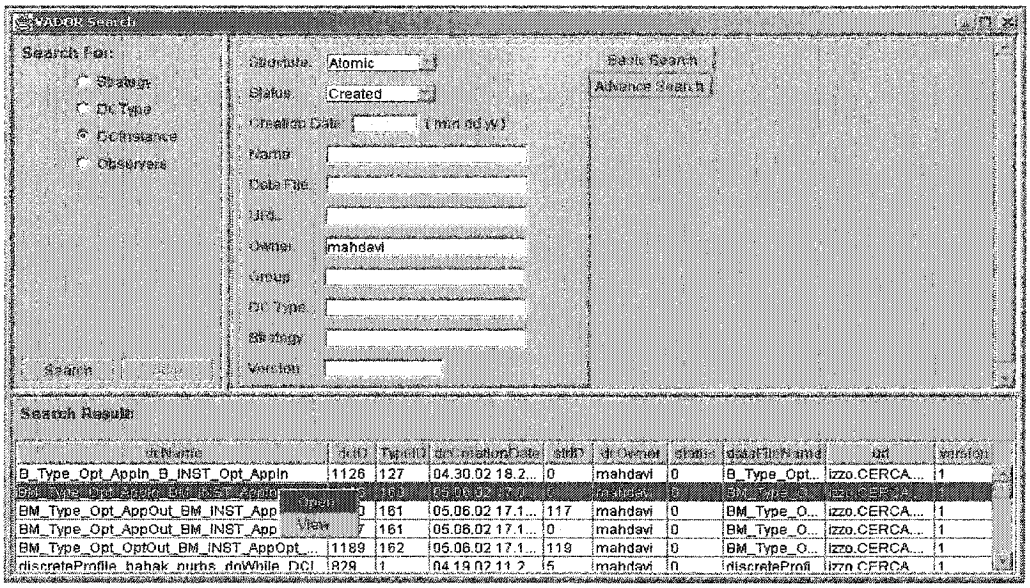


Figure 7: VADOR Search

3.6.5.2 DBExplorer

The DBExplorer is a system administration tool application (VADOR Admin. GUI), permitting the VADOR administrators communicate directly with a database to perform SQL queries, create a new group or class of host, add a new user or host, and export the database. Figure 8 shows a view of DBExplorer.

Although DBExplorer works as a separate independent application, it would be possible to integrate this application within VADOR GUI and give the launching permission only to the VADOR users with admin privileges.

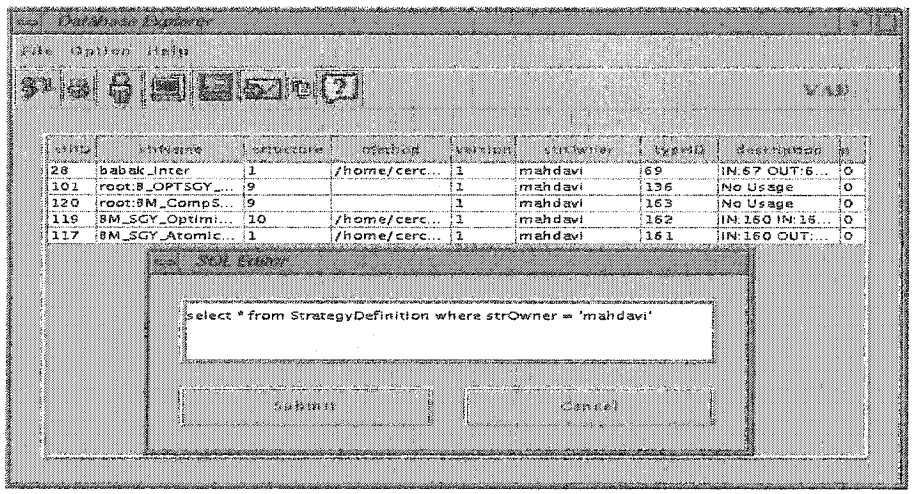


Figure 8: VADOR DBExplorer

Chapter 4: Optimization Issues

Vanderplaats [97] gives the following definition about the optimization: "The concept of optimization is basic to much of what we do in our daily lives. The desire to run a faster race, win a debate, or increase corporate profits implies a desire to do or be the best in some sense. In engineering, we wish to produce the best quality of life possible with the resources available. Thus in designing new products, we must use design tools which provide the desired results in a timely and economical fashion."

Although not precisely synonyms, the words *improvement* or *enhancement* can be two alike terms when thinking of the optimization.

This chapter discusses the optimization issues by giving examples of optimization problems, their particularity, and how they can be used within the VADOR framework.

4.1 Optimization Problem

An optimization problem in engineering includes the following parameters:

- **Function:** Function to be evaluated with an objective (e.g. minimizing or maximizing a cost function).
- **Variables:** Changeable parameters that signify a potential for change. (e.g. design variables and/or behavioural variables).

- **Constraints** (optional)**:** Limitation on solving the objective function. (e.g. limitation on the design space and/or variables). These limitations can be of any kind, mostly representable mathematically under the form of equality, inequality and side constraints.

A mathematical example of such a problem can be presented in the following form:

Minimizing:	$F(\mathbf{X})$	objective function
Where:	$\mathbf{X} = \{X1\ X2\ X3 \ldots Xn\}$	design variables
Subject to:	$g(x) \geq 0$	inequality constraints
	$h(x) = 3$	equality constraints
	$X_i^l \leq X_i \leq X_i^u \quad i = 1, m$	side constraints

A typical example of an optimization problem can be a cost function, minimizing the cost given variables and constraints. The next section discusses the problem with the programs that uses such an optimization, and how they are implemented in VADOR.

4.2 Implementation of Optimization Capabilities in VADOR

Figure 9 shows an example of an MDO optimization problem, demonstrating the data flow and dependency between various programs and stored information.

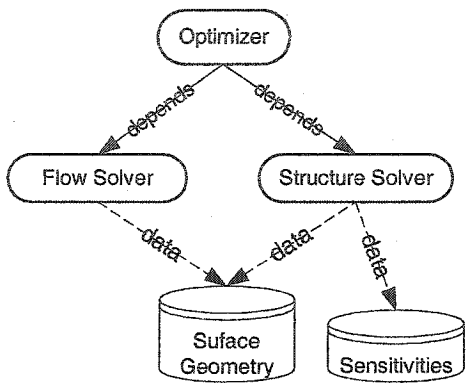
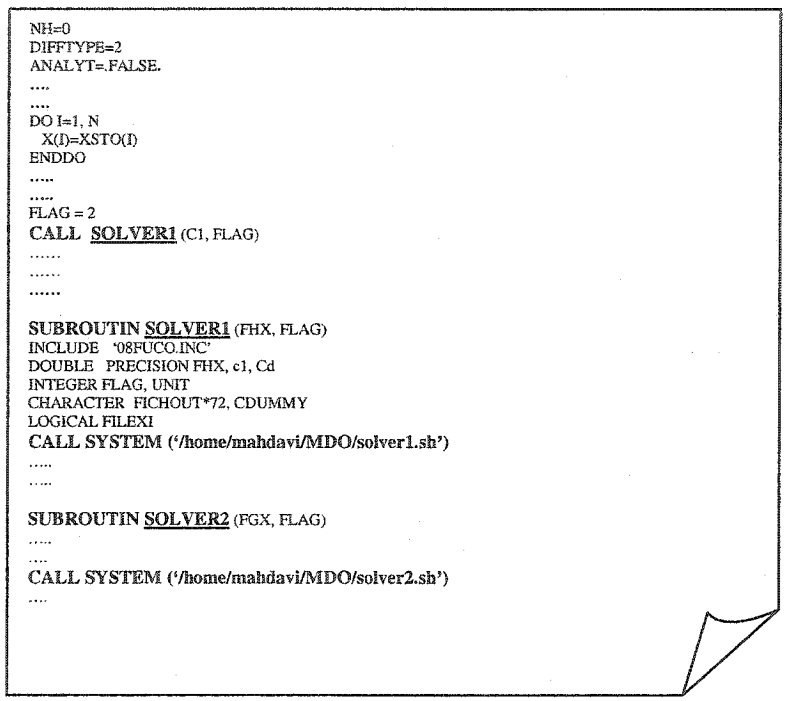


Figure 9: Example of an MDO optimization problem

In VADOR terminology, the term *Solver* is used to describe an appraisable function, namely the program that implements such an objective function. On the other hand, the term *Optimizer* is used for a program that needs to use one or more solver programs during its execution. The OptimizationStrategy is created in order to enclose an optimization process design consisting of an optimizer program encapsulated in OptimizerStrategy and the solver(s), which can be encapsulated using a normal atomic StrategyComponent(s). To understand why there is a need for such a specific StrategyComponent, it is important to recognize the difference between an OptimizerStrategy and other types of Strategies, and this becomes clear particularly in the course of the execution. The VADOR Executive, once it traverses a composite Strategy tree structure and launches every atomic node by delegating to the chosen CPU servers, does not expect any kind of interaction with the launched Strategies, rather it will be

notified by CPU Server on the success or failure of the execution, since all the encapsulated programs or scripts in Strategies have all the necessary input and output parameters in order to terminate their execution until the end. Figure 10 shows some part of a real optimizer program as example. As it can be seen, the



```
NH=0
DIFFTYPE=2
ANALYT=.FALSE.
....
....
DO I=1, N
  X(I)=XSTO(I)
ENDDO
.....
.....
FLAG = 2
CALL  SOLVER1 (C1, FLAG)
.....
.....
.....

SUBROUTIN SOLVER1 (FHX, FLAG)
INCLUDE   '08FUCO.INC'
DOUBLE   PRECISION FHX, c1, Cd
INTEGER FLAG, UNIT
CHARACTER  FICHOUT*72, CDUMMY
LOGICAL FILEXI
CALL SYSTEM ('/home/mahdavi/MDO/solver1.sh')
.....
.....

SUBROUTIN SOLVER2 (FGX, FLAG)
.....
....
CALL SYSTEM ('/home/mahdavi/MDO/solver2.sh')
....
```

Figure 10: Example of an optimizer code

program calls a solver, which in turn carries it out by making a hard-coded CallSystem for execution of the solver. It should be clear that such a hard-coded program cannot be encapsulated within a normal atomic StrategyComponent, which aims to bring the flexibility and reusability into the framework. The location of the executable programs or script may change occasionally, which results in the failure of the whole created process.

The only way to encapsulate these kinds of programs within a StrategyComponent is to replace these CallSystems by a mechanism that

permits calling to the solvers take place under the control of the VADOR system, more precisely the Executive server. The OptimzerStrategy is created as a part of solution for this particular case. An OptimizerStrategy on its path of the execution, needs the result of a solver (or solvers) in many different points of the program execution. In this case, it should be able to communicate with VADOR Executive and explicitly request the execution of the solver(s) Strategy. VADOR Executive, after the reception of such a request, will launch the solver program, encapsulated within a Strategy. Figure 11 illustrates how this mechanism is implemented in VADOR.

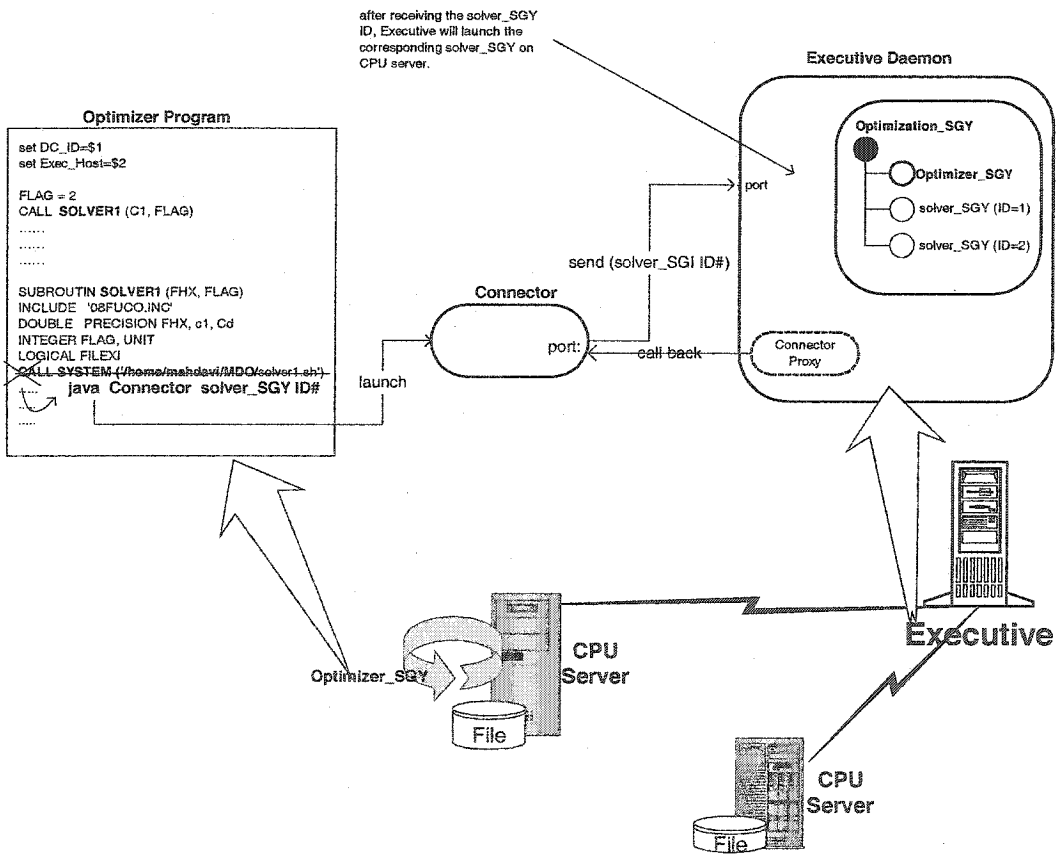


Figure 11: OptimizerSGY calls a solverSGY via Connector

An OptimizationStrategy consists of an Optimizer, and one or more solver Strategy. The execution of this Strategy starts by launching the execution of the optimizer part. The invoking CallSystems are replaced by calling to the Connector routine program written specially for establishing the connection between running OptimizerStrategy and the Executive daemon. Via this connector, the optimizer program can call a given solver Strategy by providing its ID. This is because Strategies can have the same names, but IDs are unique. Although, this may not sound very intuitive at first, the procedure is very similar to any usual programming, using a high level computer language. Most of the time, an external program or an internal function is written; or at least, it is given a name; then, this name is used to make a call to the program or the function in the main program. Here, OptimizationStrategy and obligatory all its children are first created, then the main optimizer program is completed by calling the sought solver Strategies, using their IDs instead of names.

Chapter 5: Case Studies

From the start of the VADOR project until now, many different processes have been implemented using the VADOR framework. In almost all the cases, these processes were the actual processes used in various engineering departments at Bombardier Aerospace as a part of whole big process, aiming ultimately to build aircrafts. These cases permit initially, using the VADOR prototype, to study the feasibility of the system and choose the appropriate technologies. Additional cases implemented after starting the new VADOR framework allowed not only the evaluation of the framework and verification of its specification, but also measurement of its performance and its appreciation level. Although most of these cases have been recreated and used at CERCA, today there are some processes that are only implemented and used at Bombardier. This actual use of the VADOR at Bombardier provides more accurate evaluation of the system.

The following sections describe and provide detail about some of the projects, built and used within VADOR framework at CERCA and/or Bombardier Aerospace. Although the creation of atomic and composite Data and Strategy Components are not explained, Appendix C provides a good example of how these flowcharts can be mapped into Data and Strategy components.

5.1 Damage Tolerance Analysis

Damage Tolerance Analysis (DTA) is one of the first cases implemented within the VADOR prototype. DTA is a process based on crack propagation and

residual strength analysis. One of the mission of this project was the autoimmunization of the process shown in Figure 12 [99].

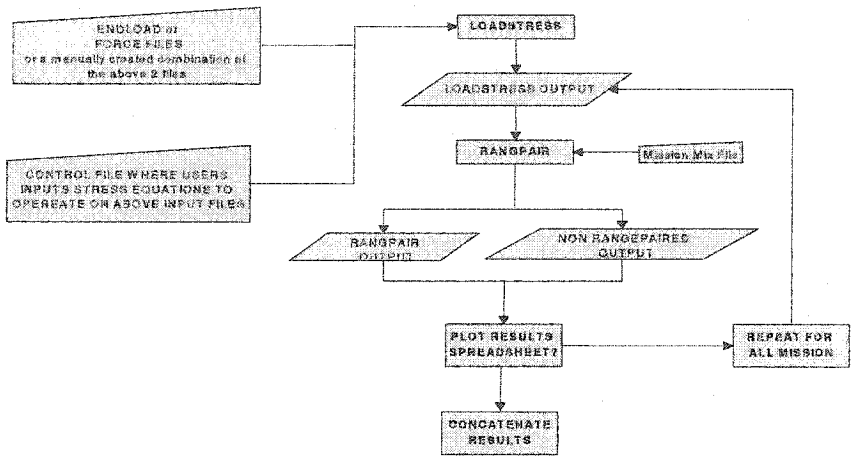


Figure 12: Initial DTA process

The *Loadstress* program pre-processes the loads for the spectrum generation computer program *Rangpair.* The *Loadstress* program selects the loads from a file and performs the specified mathematical operations. It needs two input files:

1. *control*, which contains the instructions and a list of load cases.
2. *load* , which contains all the FEM post-processed loads, produced by the Xpost program post-processing NASTRAN outputs.

The *Loadstress* program reads the loads and performs a sequence of operations on them for each load case. These mathematical operations are specified in the control file in reverse-polish notation.

A file containing the results is generated after the run of *Loadstress.* This result file must be edited to add the path of *MissionDefinition* file used to apply a factor on the fatigue unit load cases and superimpose the effects of many load

cases assumed to occur at the same time during a flight segment. The output file and the *MissionDefinition* file are used as inputs to run *Rangpair* which generates a range-pair-ranged load spectra at any particular location on an aircraft. Two output files are produced by the *Rangpair* program:

1. *range-paired*: Contains the range-paired results.
2. *non-range-paired: Contains* containing the initial spectrum.

The last step of the process is to display the spectrum in a plot tool and repeat the range-pair-range counting technique for all the different *MissionDefinition* files. This process builds the stress history for one stress component at a time as required usually. However in some cases, it is required to calculate the stress history for several components and to combine them into one single component. Figure 13 shows the process steps as it has been implemented within VADOR.

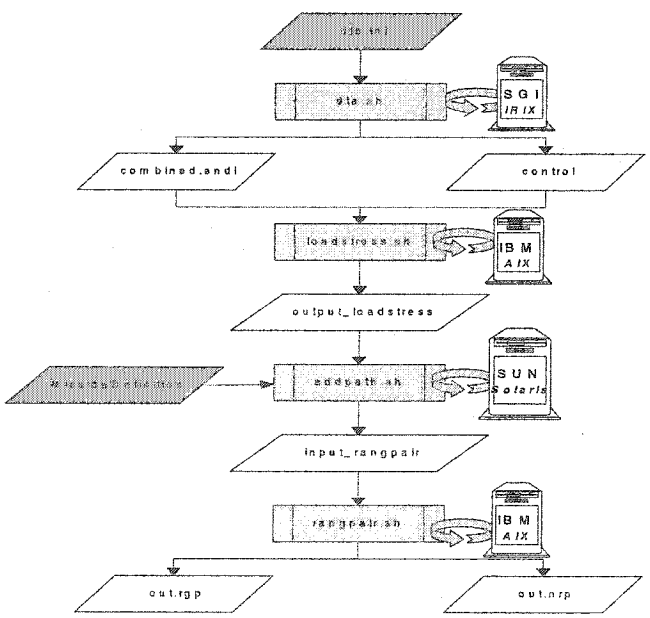


Figure 13: DTA process implemented within VADOR

A small custom GUI had to be build in order to capture user entries, including the stress equation in a normal form. In fact, the tedious part of the process, which is also the most prone to human errors, is the generation of the *load* file; the residual part of the different files containing the stress equation operands. The stress equation can be written in a regular notation using custom GUI box, then after parsing, it is converted to a reverse-polish notation and inserted in the introduction block of the *control* file (before, most of these jobs were done using manual procedures such as cut-and-paste). The *dta.ini* file is created by the custom GUI, while *dta.sh* and *addpath.sh* scripts had to be written in order to fully automate the flow of the process.

5.2 Airfoil Shape Optimization

An optimized Non-Uniform Rational B-Spline (NURBS) geometrical representation for wing aerodynamic design is used in this example. To solve this problem, an approximation of an existing wing profile (Bombardier-Canadair airfoil and the positions of its control points) is needed which implies an optimization problem and discretization. The approximation approach consists in finding an initial guess for the approximation and then optimizing the positions and weights of the control points. For the optimization, the gradient-based BFGS method is used with an objective function evaluating the average error:

$$Err_{moy} = \frac{1}{n}\sum_{k=1}^{n} d_k$$

and the maximal approximation error :

$$Err_{max} = \max \{d_k\} \; 1 \leq k \leq n$$

The objective function is then built as a combination of both errors :

$$F(X) = Err_{moy} + 2 * Err_{max}$$

where X is the design variables vector containing the positions and weights of the approximation control points. The choice of the initial guess is crucial if the approximation method is to be reliable and efficient. Indeed, the objective function of such a problem is strongly non-linear, so the optimizer is most likely to diverge, or to converge towards a local minimum, if the initial guess is too far from the solution. The optimization process is depicted in Figure 14.

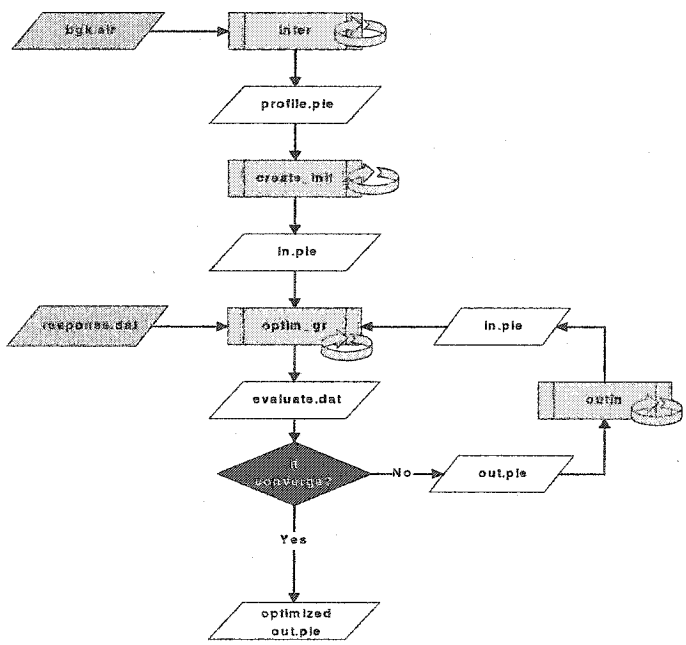


Figure 14: Airfoil shape optimization process

Brief descriptions of programs:

- The data file *bgk.air* contains the wing profile in the form of a set of points.
- The *inter* program interpolates the profile using B-spline with the same number of control points. The order of the B-spline is specified by the

user, and if it is chosen to be 2, the program simply transforms a *bgk(.air)* format file to *pie* format file, here *profile.pie*.

- The *create_init* program creates an initial file *in.pie* for the optimization of the NURBS representation of the wing profile described in *profile.pie*.
- The *optim_gr* program performs the optimization of the NURBS representation of the wing profile using BFGS method. It needs user interaction to set the optimization variables. It generates two dataset files, one containing the optimized profile *out.pie* and the other the errors of the approximation *evalue.dat*.
- The boolean program *if converge?*, can determine if the optimization has converged or not after a given number of iterations. If not, another optimization loop is started.
- The *outin* program converts the optimized profile *out.pie* to the input format of the optimizer, if the process has not converged.

5.3 Bend and Twist

The Bend and Twist process (Figure 15) is an iterative process used to compute the wing deformations for some specified flight conditions. The starting point is a

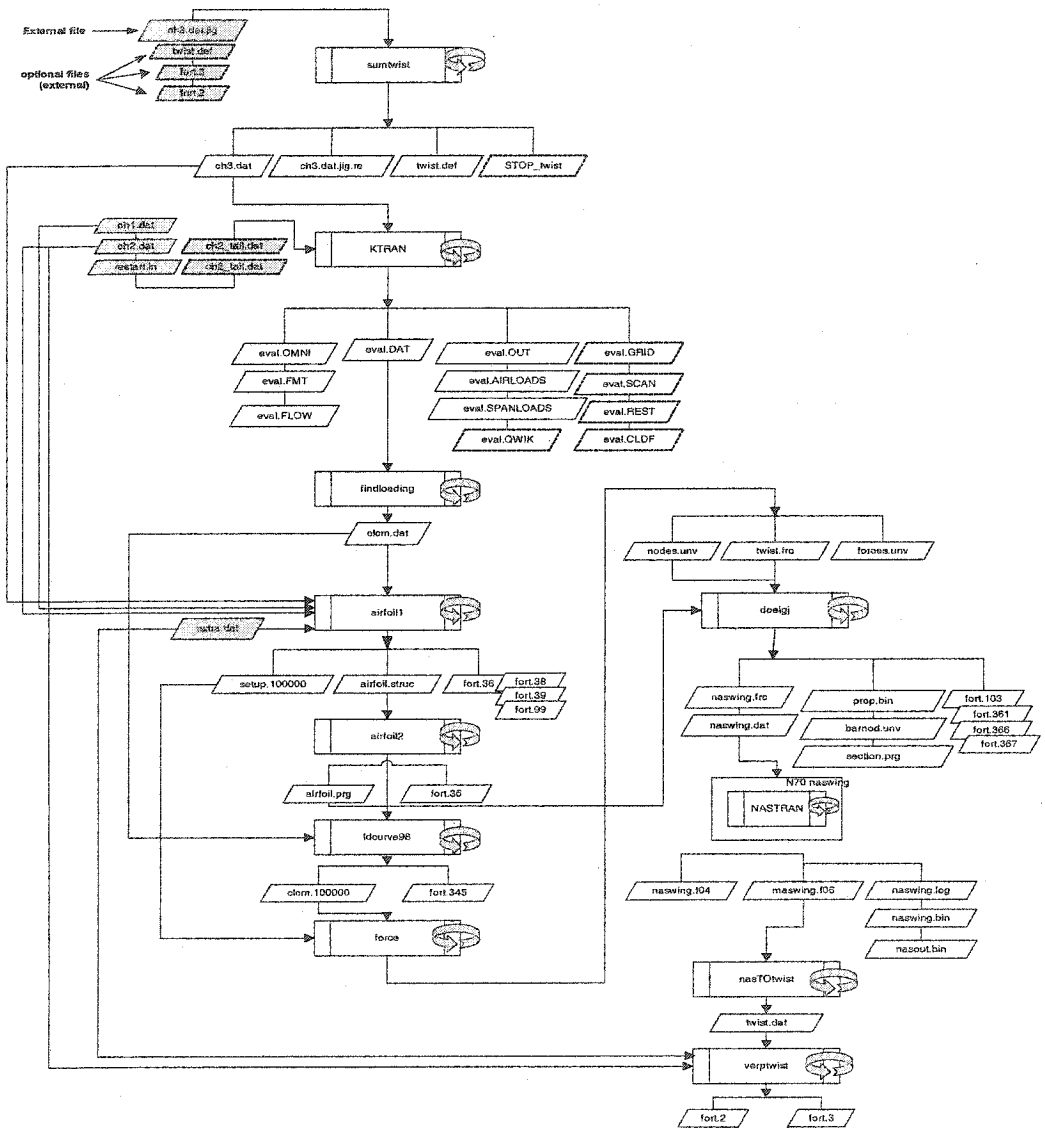


Figure 15: Bend and Twist process

wing *jig* model. The *jig* model represents the geometry of the wing as it is built. Under the aerodynamic loads encountered in flight, the wing bends and twists to take the flight geometry. Unlike the *jig* geometry that is unique, a flight geometry

exists for each set of flight conditions simulated. The bend and twist process, after convergence, returns the vertical and twist deflections of the wing and therefore, the desired flight geometry. The Bend and Twist Process implemented in VADOR is used for a wind-tunnel model. A similar process, involving a few different subroutines for the computation of the structural properties, exists for the real wings.

Brief descriptions of programs:

- *sumtwist* : Adds the twist deflection computed in the previous iteration to the jig twist to produce the geometry given to KTRAN.
- *KTRAN* : Computes the flow and the aerodynamic loads on the model.
- *findloading* : Extracts the wing loading from the KTRAN solution.
- *airfoil1, airfoil2* : Calculates the geometric and structural properties.
- *ldcurve98, force, doeigj* : Pre-processing for NASTRAN.
- *NASTRAN* : Computes the wing deflections in response to the loads applied.
- *nasTOtwist* : Extracts the twist and vertical deflections from the NASTRAN output file.
- *verptwist* : Reformats the output data in a *verp* (plotting software) format.

5.4 PRE-NSU3D

The PRE-NSU3D process is used to convert a grid file produced by the ICEM software into a format that can be easily read by the NSU3D code. The process involves different utilities, as well as a recompilation step. The complete process needs also to be run on two architectures, namely IRIX and CRAY machines. Figure 16 presents PRE-NSU3D in form of the flowchart.

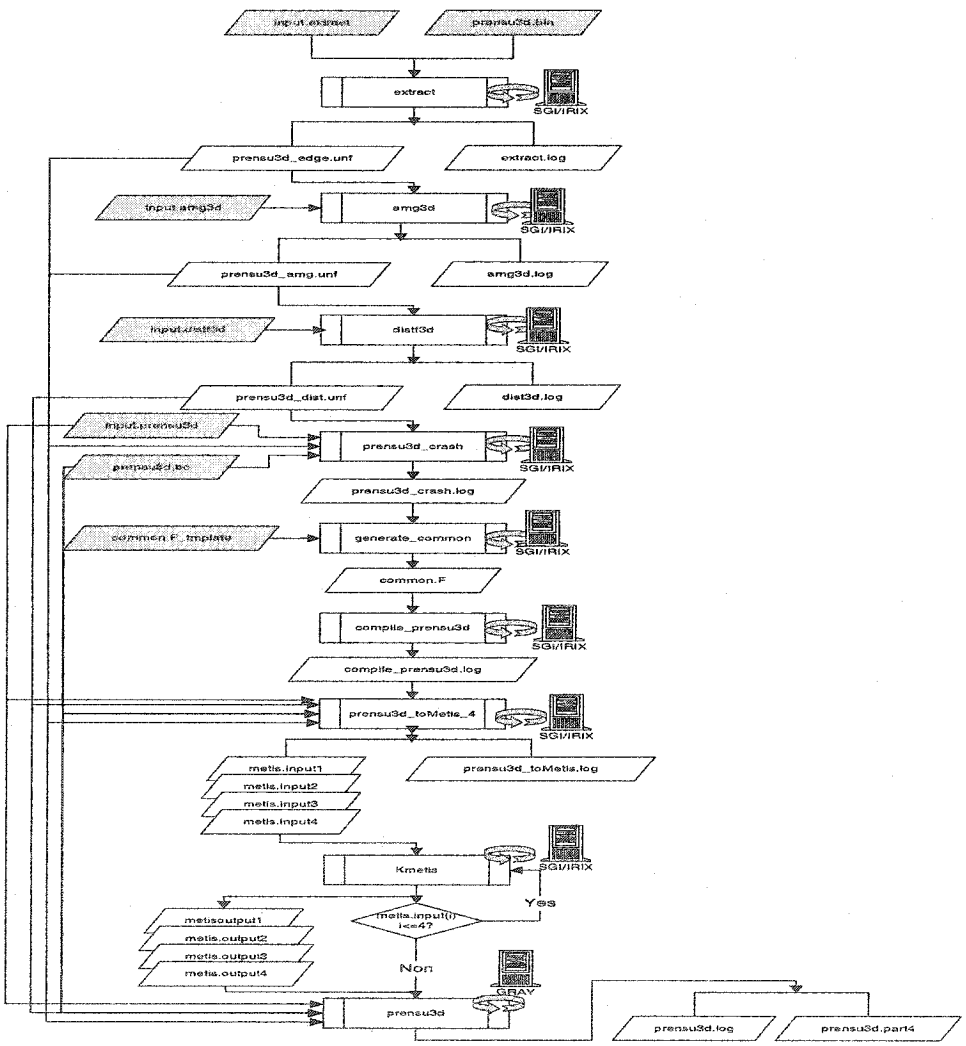


Figure 16: PRE-NSU3D process

Brief descriptions of programs:

- *extract* : Converts a *.bin file (obtained from ICEM) into an *_edge.unf file. The *_edge.unf file contains the list of edges and the edge coefficients.
- *amg3d* : Pre-processor which builds the coarse multi-grid levels for the agglomeration multi-grid algorithm employed by NSU3D.
- *distf3d* : Pre-processor which computes the distance function required by certain turbulent models in the flow solver.
- *prensu3d_crash* : A version of the prensu3d program that was compiled with minimal arrays sizes. The lack of memory makes the program crash.
- *generate_common* : From the log file created with the crash of prensu3d, this program generates a Fortran file containing the appropriate sizes for all the arrays.
- *compile_prensu3d* : Compiles prensu3d using the Fortran file generated in the preceding step.
- *prensu3d_toMetis_4* : General pre-processor. The first call to prensu3d creates the output files for the Partitioner.
- *kmetis* : Grid partitioner. Creates the partitions.
- *prensu3d* : Generates the fully partitioned output necessary for the NSU3D flow solver. Final step before running NSU3D. This execution is done on the CRAY machine.

5.5 Observation Results on Implemented Case Studies

Despite of the differences between these implemented processes, some common advantages and improvements of using the VADOR framework versus the actual in-use techniques have been experienced. These are:

- Fully automation of the process – no more manual interaction of the user is needed. Furthermore, the user can work on different tasks while VADOR is executing the process. This automation benefit alone, can already save a lot of time (therefore money) and justify the use of such a tool. In fact, the DTA study shows that about 35% of time-saving was attributed to the automatization of the process compared to the previous in-use manual procedure [99].
- Visual representation of the process – this provides a better global view of the process and involved data. The user sees what has been done and what remains to be done.
- Visual representation of the process execution – the actual status of the execution is known, so the user is alerted to the task being executed and the data already created.
- Traceable records of the performed tasks – every piece of data can be traced (and reproduced if necessary). In addition, add-in applications can be used to open the traced data for editing or visualization purpose.
- Lower risk of errors – programs have inputs and outputs with predefined types. This reduces the danger of executing a program with a wrong input

file. Moreover, the user cannot delete by mistake a file that is linked to other files.

- More reusable process – standardization of the data and process into components promote the reusability of the process.
- Easier tracking of the work of others – the process is somehow documented, so it is possible for others to see what has been done by other people in the project.
- Refined search – by classification of the system, and providing an advanced search tool, enriched searches can be performed on data and components in the system.
- File duplication prevention – unnecessary or erroneous duplication of files is prevented or at least minimized.
- Easier training of new staff – already created process is easier to understand for the new unfamiliar project member. Once understood, the same concept, uniform appearance, and approach are used for creating, using, or understanding the other processes.
- More conventional practice – using different user-friendly common widgets such as browser, permits an easier and more intuitive practice at work.
- Enforcement of the standard practices.

Chapter 6: Future Work

VADOR constantly improves and a list of planned enhancement is always prepared to ensure that this framework traverses its path towards an increasing level of maturity. In the next phase of development, the incorporation of the Web technologies is envisaged. Usage of the Extensible Markup Language (XML) [100] can provide a common information format that can be exploited in different forms. Every component in the system is linked to an XML file, which keeps the record of the course of the execution, and the eventual errors in XML format. This can improve the debugging aspect of the system. Later, it would be also possible to define components in XML format and export them to the VADOR database.

The execution of the cross-organization distant process that involved two Bombardier branches located in two different distant cities is emphasized. The totally different administration and account management even within the same organization creates more server security concerns. It is important to shift the socket-based communication into a higher level form of network communication where standard protocols such as Secure Sockets Layer (SSL) and Transport Layer Security (TLS) can be employed. This is also crucial for keeping the integrity of the account of all the VADOR users, even within the same organization location. The deliberate or inadvertent kind of not-trusted Strategy execution is thus not possible, creating a safer environment and avoiding catastrophes.

A bypass solution for encapsulating discipline codes on the CRAY [101] machines is also undergoing. There is no available JVM for these supercomputers at this moment, making it impossible for either the VADOR servers or VADOR GUI to run on these super machines. This means a user cannot even browse a directory on CRAY computers in order to enclose discipline codes within a StrategyComponent.

The use of parallel computing techniques still remains challenging. VADOR should be assured that parallel processes are truly done in parallel, at the same time, and on different machines or processors. This is different from pseudo-parallelism, where processes are launched simultaneously on the machines equipped with only one processor, without being sure that two parallel sub-processes are not threaded on the same machine. Although early advanced frameworks took advantage of conventional standards such as PVM and MPI [102], the new trend in research towards new ways of parallelism in the broader dimension, using technologies such as CORBA and Grids, [60][61][62] are certainly worthy.

The use of the Artificial Intelligence (AI) techniques (such as efforts done on OASIS system) are practically missing in the development of all the actual advanced frameworks. AI techniques can definitely bring enhancement and improvement in the automation of the design. By the acquisition of information and with rules for using the information, and using the rules to reach approximate or definite conclusions, it is possible to move from some existing knowledge-based frameworks towards a more expert kind of system in which the use of

information is more dynamic. For instance, applying search and machine learning techniques can help to propose the best performance profile based on the past optimization occurrence on similar problems

Subtly in the same direction, undergoing study of the Enterprise Resource Planning (ERP) tools provided by leader companies in this field, such as SAP [108][109] , BaaN [110], Oracle [111], PeopleSoft [112], and J. D. Edwards [113] can certainly help to create new modules within the VADOR framework that assist engineers and design, process, risk and change managers by giving them the new facet of the VADOR utility. Although the initial basic idea about these ERP tools was initially to provide customers with the ability to interact with a common corporate database for a comprehensive range of assembled applications, today these tools manage the important parts of any business, from finance to human resources, including product planning, parts purchasing, inventory management, interacting with suppliers, customer service, and order tracking. In fact, the ERP approach shares some similarity change paradigm that VADOR also tries to bring into the engineering work environment. The approaches are information- and process-oriented (bring attention from data to process), trying to integrate these processes in order to bring a manufacturing performance improvement, using the business or industry change effort strategy. ERP tools also help industrial operations optimize their enterprise performance strategies in the knowledge-driven environment, with its ever-increasing demands for information, integration, and collaboration. After all, MDO also aims at economic competitiveness, a balanced product performance encompassing

manufacturing, life cycle issues, design process timetable, and economics. Historically, the integration of ERP tools was not very successful in the engineering-oriented organization contrary to any other kind of organization. Using an engineering-oriented tool that integrates partially or totally the fundamental engineering-management modules can possibly bring new accomplishments not yet achieved in this field. An example could be the creation of a new module within the VADOR framework that collects the existing dispersed information about processes and people involved in a given project, and provides the manager with the appropriate precise information on the progression of the particular project, including detailed data on the advancement of each sub-process done individually by the responsible engineers. Managers could then calculate the risks and apply all the necessary changes to the process.

Chapter 7: Summary and Conclusion

MDO definition, its value, particularly in the aerospace industry, and the motivations for the design of a problem solving environment where MDO concepts can be practiced, were briefly described in Chapter 1.

In Chapter 2, a short history of the MDO origin, its objectives and the multiple open research areas on this subject were presented. Moreover, it was explained how an MDO framework is defined and what are the requirements for such a problem solver environment. This was followed by a concise description of all the current and former efforts for creating such a framework, having the goal of allowing the MDO practices. Most of the presented frameworks were developed only for research purposes, such as FIDO and IMAGE. Some of these projects gave way to new projects using the acquired experience from previous work, or they were simply discontinued. Many of these R&D efforts have been obviously done by universities, although NASA remains the main organism interested by the subject, considering the fact that most of the R&D realized by universities were also financially supported by NASA. Many of these frameworks are case specifics, designed only for solving particular MDO problems and they may not suitable for generic MDO practices. For instance, ProFES is intended for performing RBMDO problems. Stiff competition seems not to exist among the builders of the remaining generic MDO commercial frameworks, since users can study their needs and restrictions, in order to easily find the tool that suits them best. It is not surprising that the organizations that do not find any of these

commercial tools fitting in well with their working environment or their current technologies try to command a customized system that takes into consideration the particularities of the industry.

Chapter 3, presented VADOR, a generic MDO framework, where its specifications attempt to meet the requirements of any ideal MDO framework and one of its specific objective is to bring MDO capability practices within the Bombardier Aerospace, considering the mythologies in practice in this aerospace industry. The critical review of the VADOR versus framework requirements demonstrated that, although this framework is still far from being an ideal framework, given the constant improvements and anticipated modifications the potential exists that a very notable and unique MDO framework will result from these efforts. Java is certainly a good choice for developing in a heterogeneous, object-oriented, component-based environment; however some promising Web facilities using Java applets seem to have some server security limitations on writing and reading files, creating a limitation on its large-scale MDO capabilities with regards to usage of a Web browser as the interface to the framework. Another difficulty was the unsolved latency problem during callback using RMI, which forced VADOR for the time being to use the socket-based communication. The only advantage of using this form of communication socket over other technologies such as RMI/CORBA is its performance in terms of speed. Socket is slightly faster than RMI or CORBA. On the other hand, the socket programming model is very primitive, using only a very low level of abstraction. A custom protocol for passing the request of service names and their arguments

should be created. Also, implementing all these multiple services causes major maintenance and programming problems. In addition, transactions, distributed method invocations, dynamic discovery and metadata support, are not supported. Furthermore, standard security protocols such as SSL and TLS which demand a higher level form of network communication cannot be employed.

The two kinds of component family in VADOR were also introduced. The first one encapsulates the data, while the second component family encapsulates programs and the design process. The advantage of this component-based approach will be even more apparent when other component-based standards such as CORBA are used. Moreover, extra effort should be made to make sure the VADOR components are also the Java components. This can be done by making sure that they fully conform to the rules of JavaBeans. Thus, the broader benefits of these true component-based environments can be explored by taking full advantage of Java programming language. The different types of design patterns implemented in VADOR were also explained. The idea is to bring more modularity and consequently more reusability of different part of the system, which is important for any complex distributed architecture such as VADOR. Different parts of the VADOR multi-tiered distributed architecture were illustrated including servers and load balancing which aims to bring the equitable distribution of workload among the VADOR CPU servers.

In Chapter 4, the concept of an optimization problem was enlightened using an example, and the technical problem of using optimizer program was explained

along with the solution for solving this impasse in the optimization capabilities of the VADOR framework.

In Chapter 5, the case study processes implemented within the VADOR framework at CERCA and Bombardier Aerospace were depicted using their respective process flowcharts. The advantage of using the VADOR framework versus the former manual techniques were enumerated, giving evidence of its sure road towards a higher level of maturity. And finally, a list of undergoing and future planned work along with subtle suggestions, were presented in Chapter 6.

The increasing complexity of the products within an economic competitive environment, accentuates the need to reduce the design cycle time. This need is even bigger in aerospace industry, where the more complex aircraft design is multidisciplinary in nature, and different disciplines involved in design, work independently of each other. In such an environment, the need to apply the MDO practices is essential. VADOR is not the first effort to create an environment enabling MDO solutions, and it will certainly not be the last. The challenges of creating an ideal MDO environment are enormous. VADOR succeeds in creating an automated engineering and time-effective environment, where seamless integration of the legacy analysis codes and the design processes are visually feasible. This MDO-enabling distributed framework gives visibility to the processes, permitting the change tracking in a design project and possibly monitoring of the progress. Any other tentative for creating a similar distributed system can take advantage of most of the concepts used in the development of this framework or avoid its faults. The multi-tiered architecture, design patterns,

component-based and object-oriented approaches are now known concepts. The key is to use these standards properly within a complex distributed system designed for large-scale engineering tasks.

Appendix A: Nomenclature

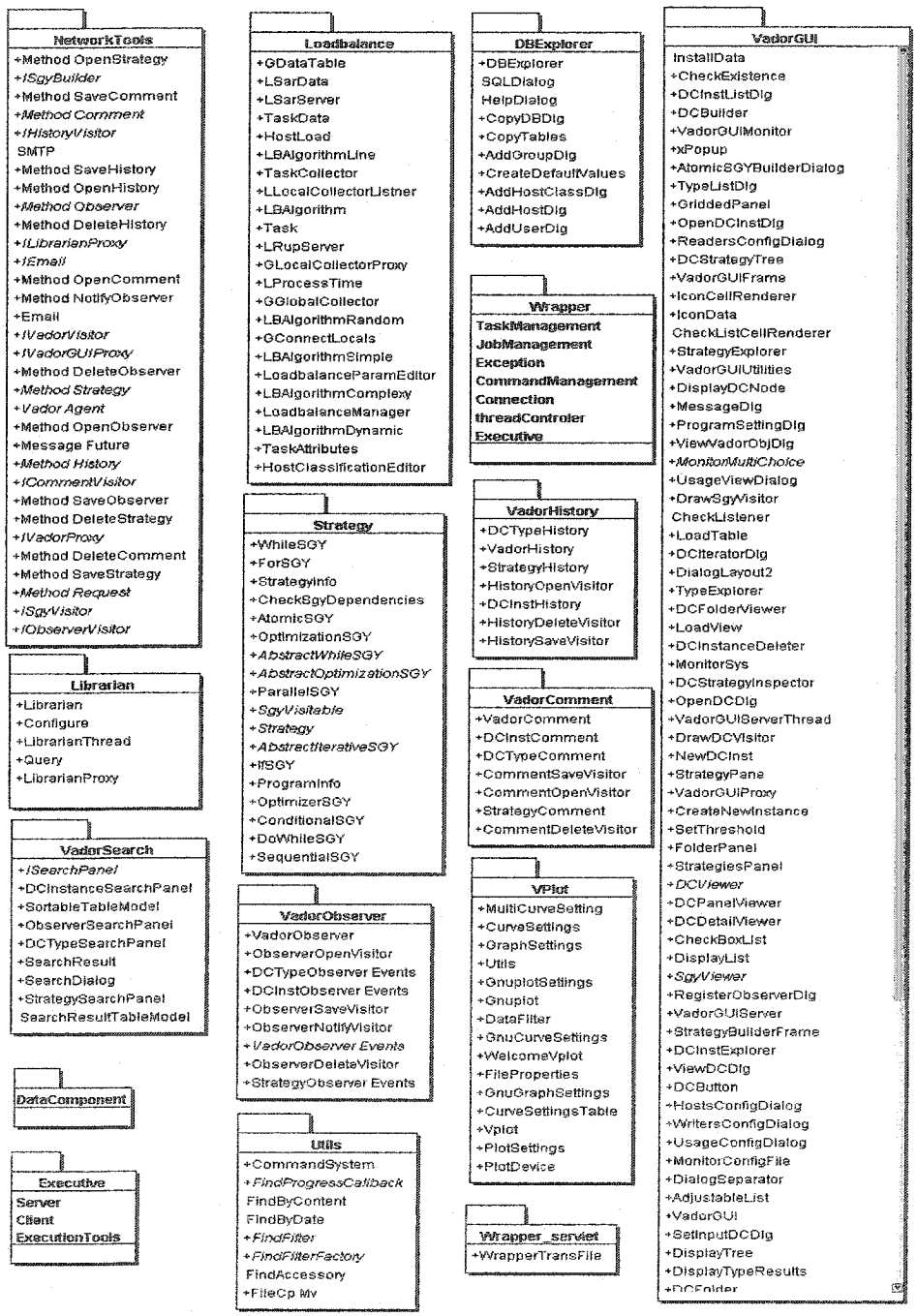
AI	Artificial Intelligence
AML	Adaptive Modeling Language
ANSI	American National Standards Institute
API	Application Program Interface
ARA	Applied Research Associates
ASDL	Aerospace Systems Design Laboratory
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAT	Component Architecture Toolkit
CATIA	Computer Aided Three-dimensional Interactive Application
CCA	Common Component Architecture
CCAT	Common Component Architecture Toolkit
CE	Concurrent Engineering
CERCA	CEntre de Recherche en Calcul Appliqué
CFD	Computational Fluid Dynamics
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CSD	Computational Structural Dynamics
CVS	Concurrent Versions System
DAKOTA	Design Analysis Kit for Optimization and Terascale Applications
DARWIN	Developmental Aeronautics Revolutionizing Wind-tunnels with Intelligent systems for Nasa
DBMS	DataBase Management System
DC	Data Component
DCOM	Distributed Component Object Model
DOE	Design Of Experiments
DTA	Damage Tolerance Analysis
EJB	Enterprise JavaBeans
ERP	Enterprise Resource Planning
FEA	Finite Element Analysis
FIDO	Framework for Interdisciplinary Design Optimization
GSE	Global Sensitivity Equation
GUI	Graphical User Interface
HPCCP	High Performance Computing and Communications Program
IMAGE	Intelligent Multidisciplinary Aircraft Generation Environment
IO (I/O)	Input Output (Input/Output)
IP	Internet Protocol
IPG	Information Power Grid
ISO	International Organization for Standardization
JDBC	Java DataBase Connectivity
JPL	Jet Propulsion Laboratory

JVM	Java Virtual Machine
LaRC	LAngley Research Center
MDICE	Multi-DIsciplinary Computing Environment
MDO	Multidisciplinary Design Optimization/Multi-Disciplinary Optimization
MDOB	Multidisciplinary Design Optimization Branch
MDOL	Multidisciplinary Design Optimization Language
MIDAS	Multidisciplinary Integrated Design Assistant for Spacecraft
MODEL	Multidisciplinary Optimization and Design Engineering Laboratory
MPI	Message Passing Interface
NASA	National Aeronautics and Space Administration
NFS	National Science Fondation
NPSS	Numerical Propulsion System Simulation
NSERC	Natural Science and Engineering Research Council
NSF	National Science Foundation
NURBS	Non-Uniform Rational B-Spline
ODBC	Open Database Connectivity
OMG	Object Management Group
OMT	Object-Modeling Technique
OS	Operating System
ProFES	PRObabilistic Finite Element System
PVM	Parallel Virtual Machine
QEM	Quality Engineering Methods
R&D	Research and Development
RBMDO	Reliability-Based MDO
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSM	Response Surface Modeling
SAP	Systems, Applications, Product
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
Tk/tcl	Tool Kit / Tool Command Language
TLS	Transport Layer Security
UML	Unified Modeling Language
URL	Uniform Resource Locator
VADOR	Virtual Aircraft Design and Optimization fRamework
VR	Virtual Reality
WICkED	Web Interface for Complex Engineering Design
XML	Extensible Markup Language

Appendix B: UML Diagrams

B1 UML Package

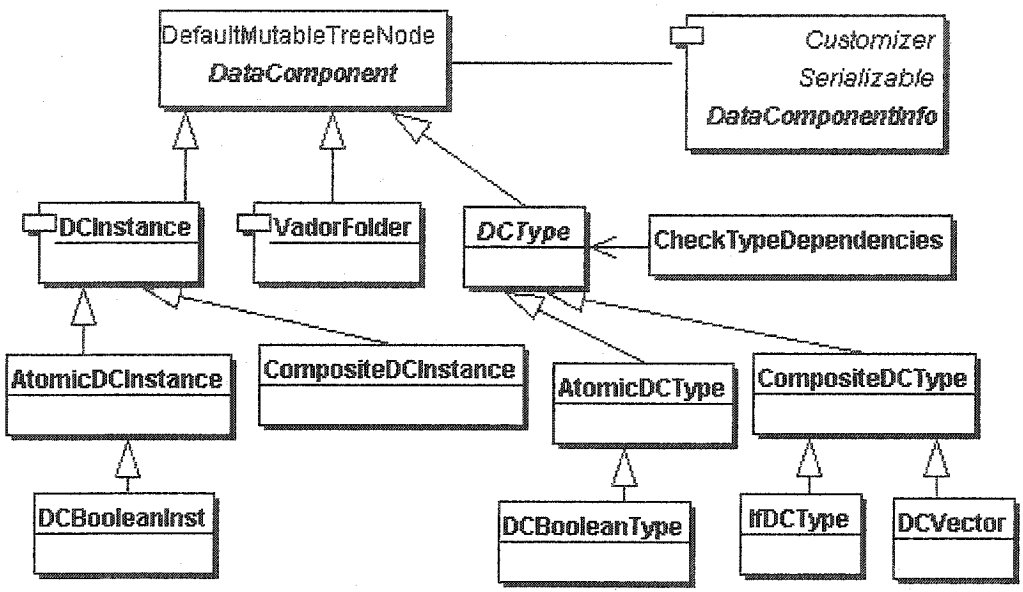
B1.1 VADOR Packages



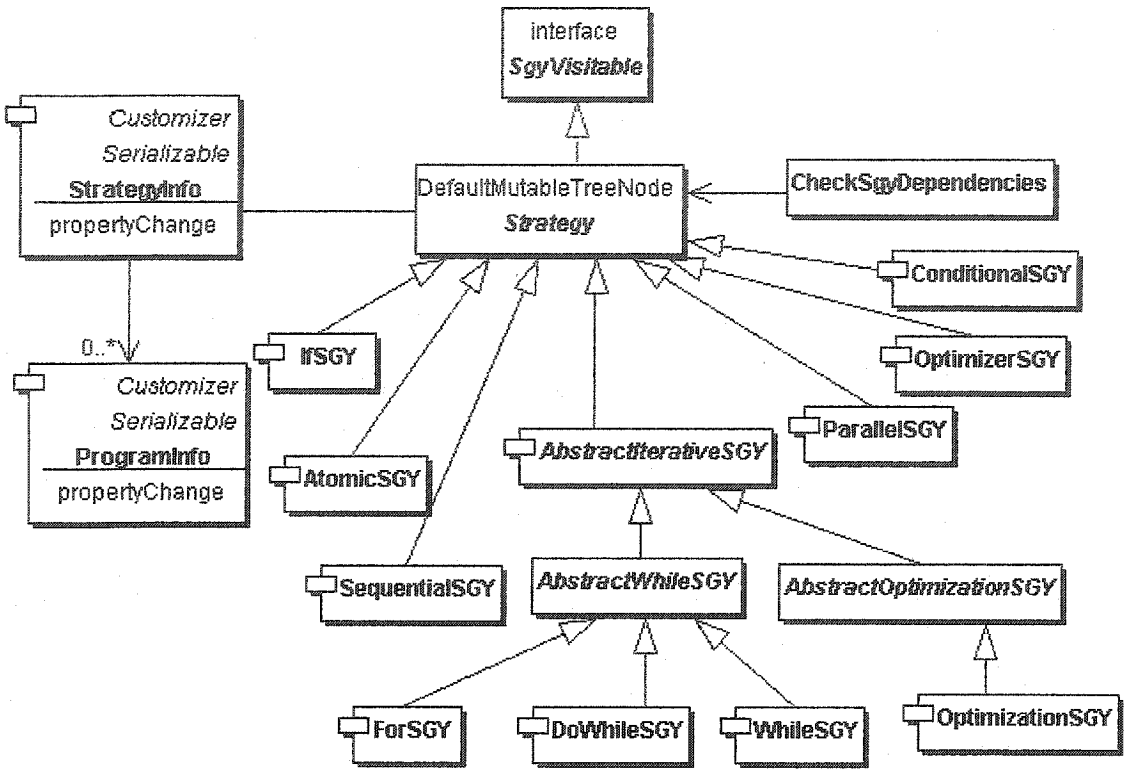
B2 UML Class Diagrams

The following diagrams show the main classes and their relationship to each other in their respective packages.

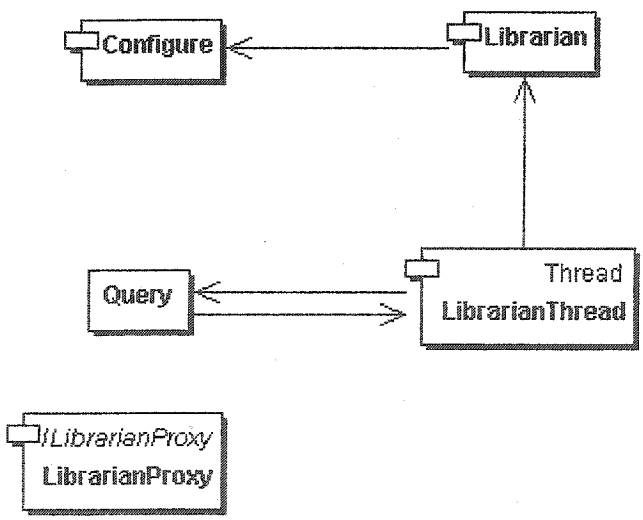
B.2.1 DataComponent Class Diagram



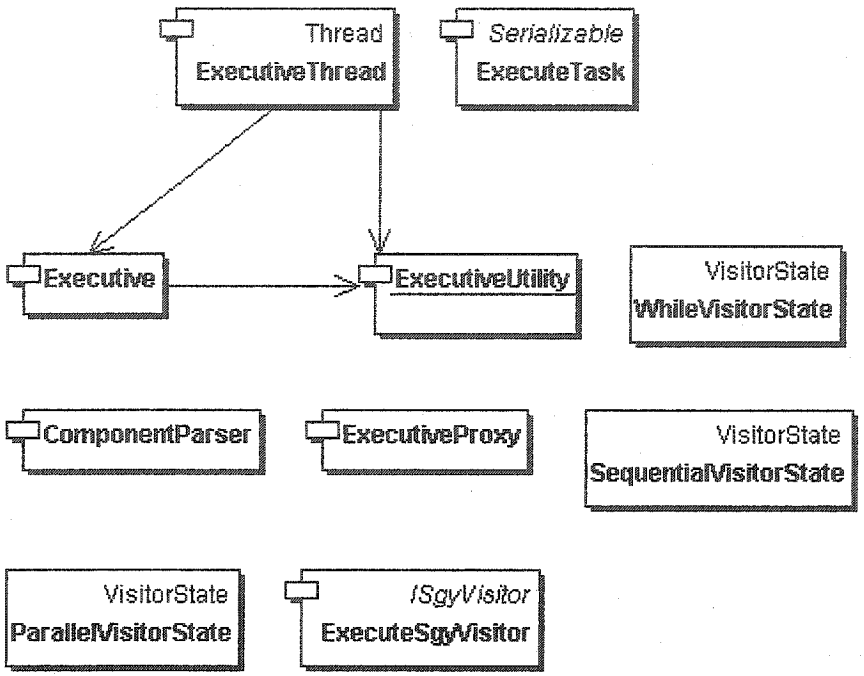
B.2.2 StrategyComponent Class Diagram



B.2.3 Librarian Class Diagram



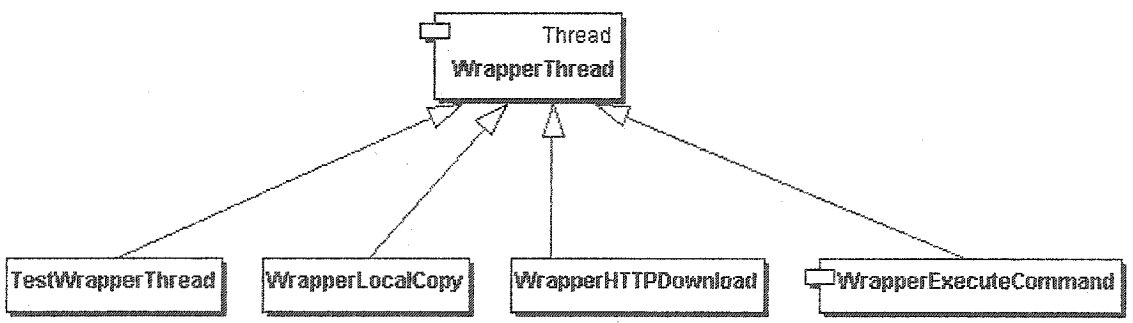
B.2.4 Executive Class Diagram



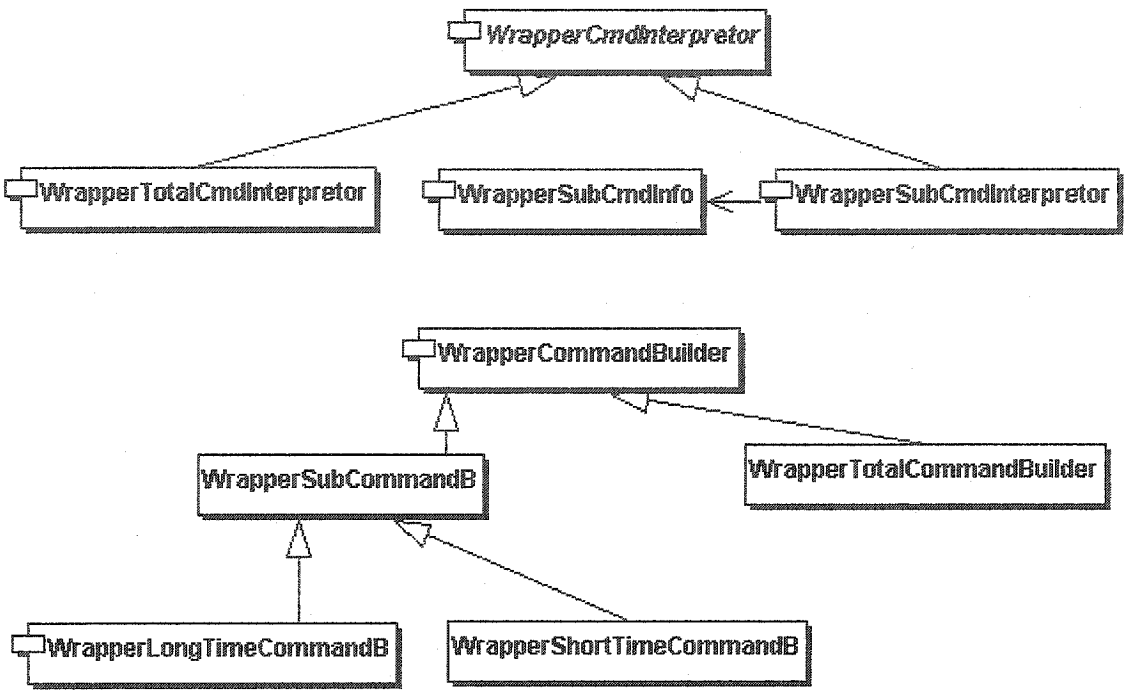
B.2.5 CPU Server (Wrapper) Class Diagram

The Wrapper package contains the classes for implementation of CPU Servers. These classes are regrouped in subpackages shown in Sections B.2.5.1 to B.2.5.5:

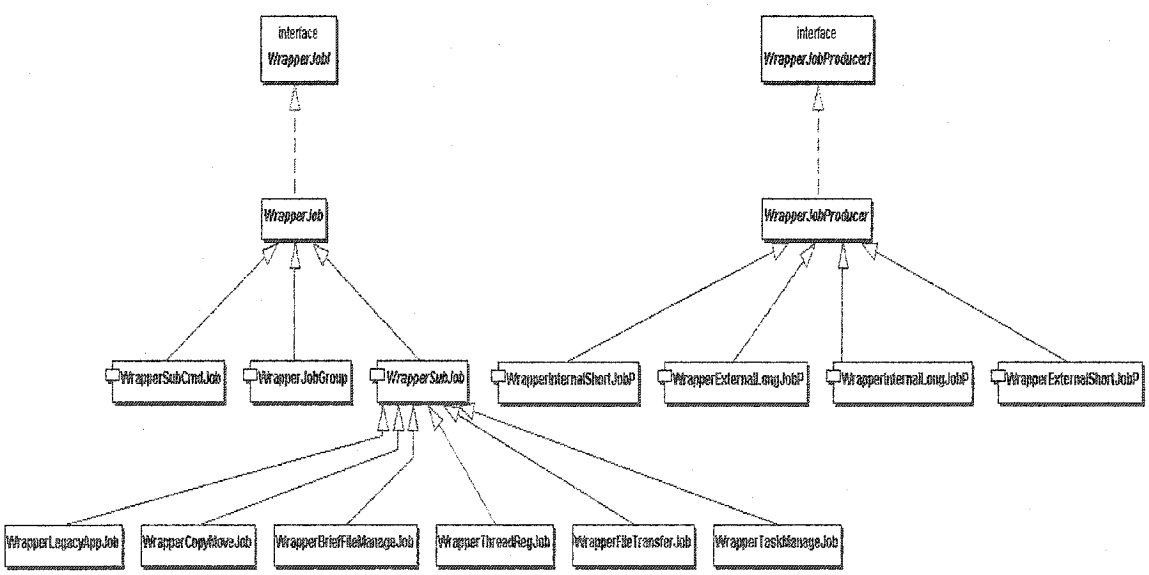
B.2.5.1 Wrapper / Thread Controller



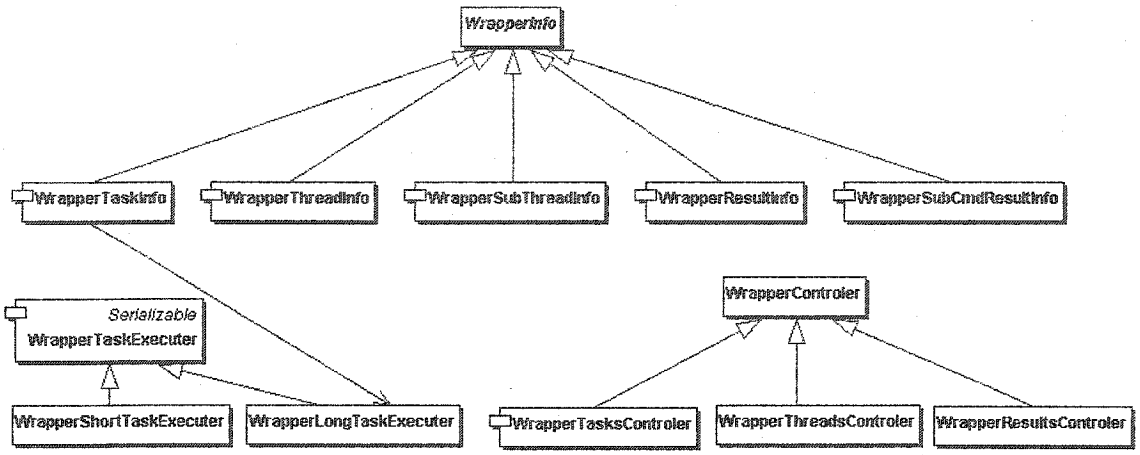
B.2.5.2 Wrapper / Command Management



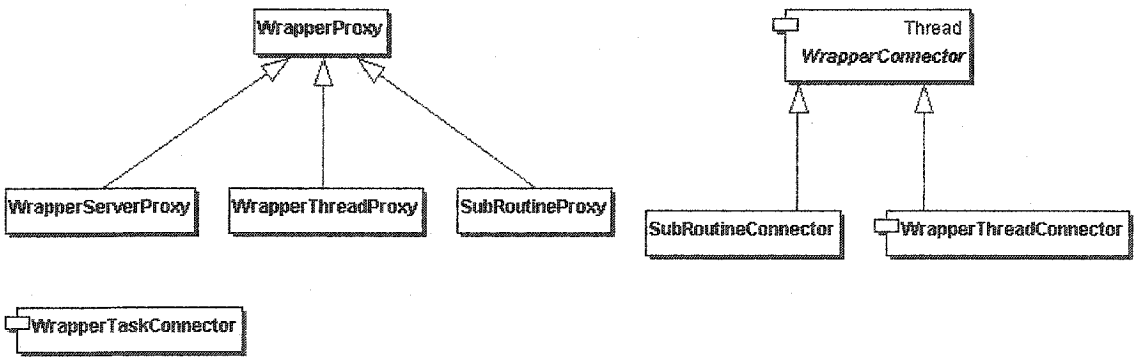
B.2.5.3 Wrapper / Job Management



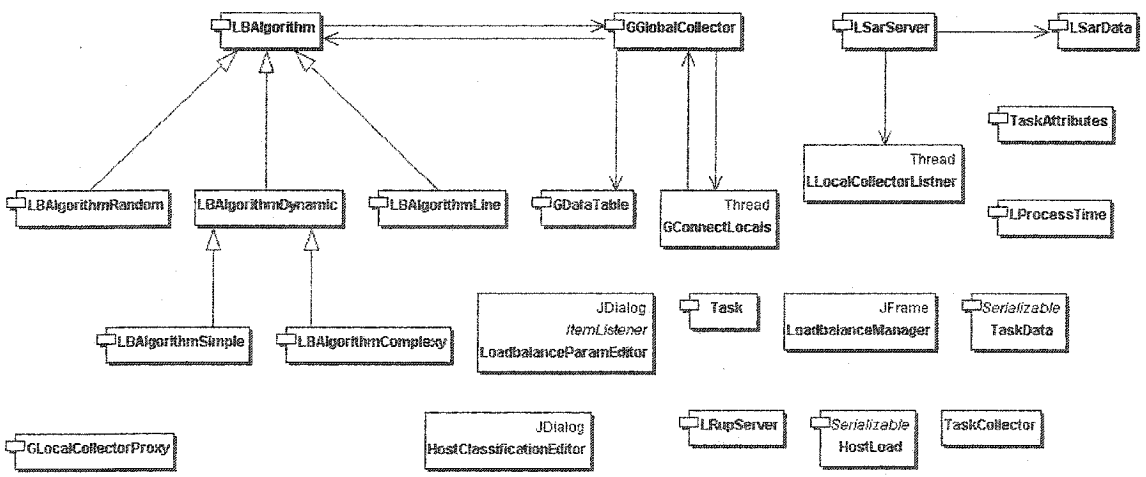
B.2.5.4 Wrapper / Task Management



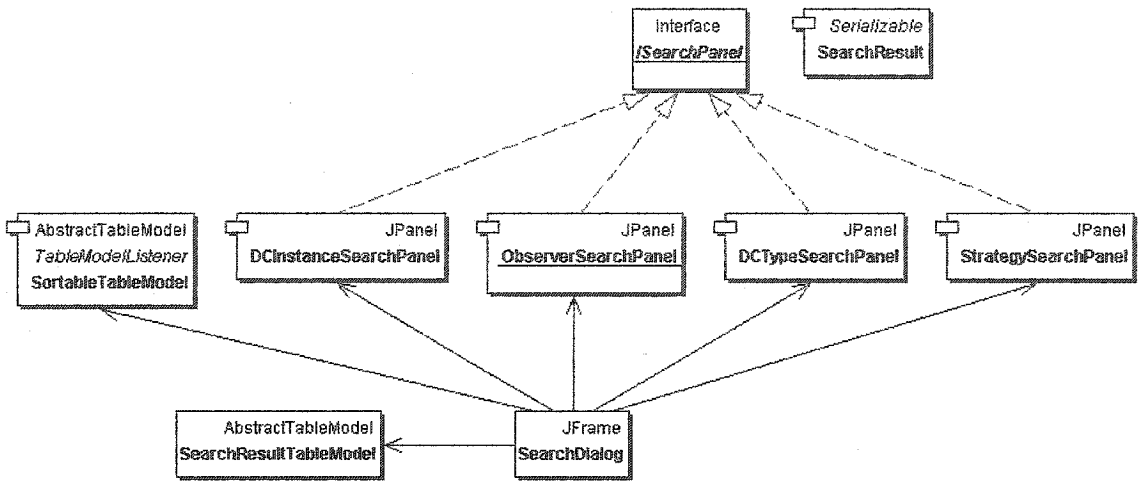
B.2.5.5 Wrapper / Connection



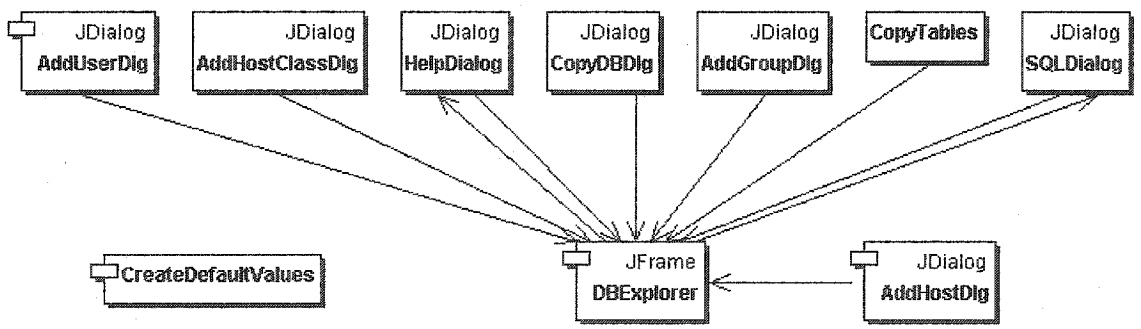
B.2.6 Loadbalance Class Diagram



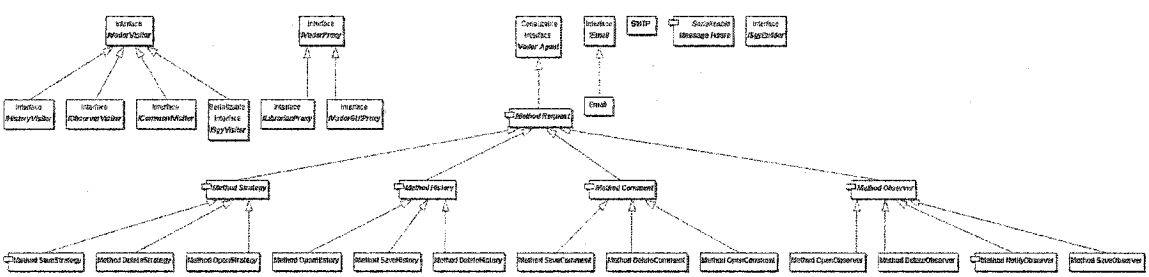
B.2.7 VADOR Search Class Diagram



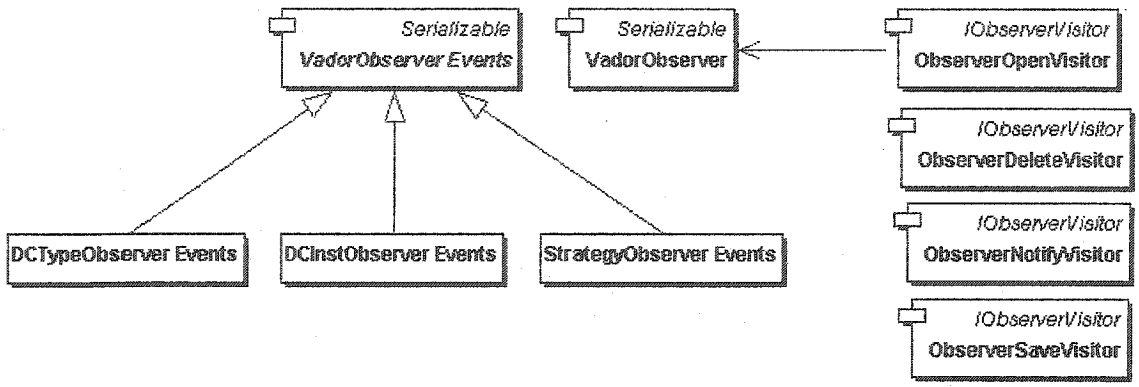
B.2.8 DBExplorer Class Diagram



B.2.9 Network Tools' Package Class Diagram

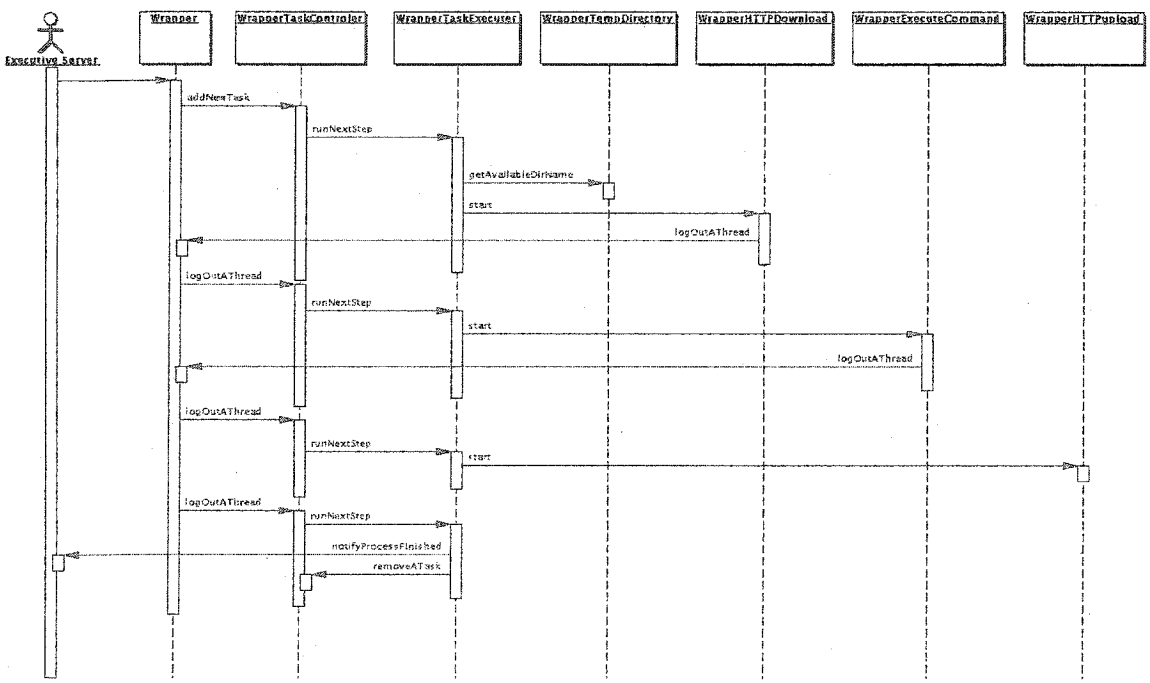


B.2.10 VADOR Observer Package Class Diagram



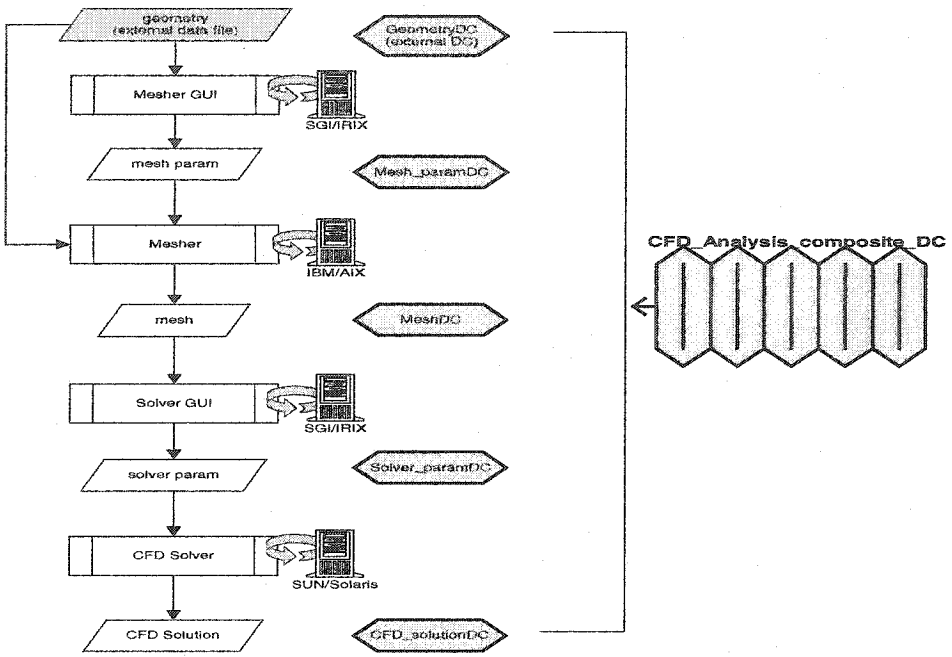
B3 UML Sequence Diagrams

B.3.1 VADOR CPU (Wrapper) Server Sequential Diagram

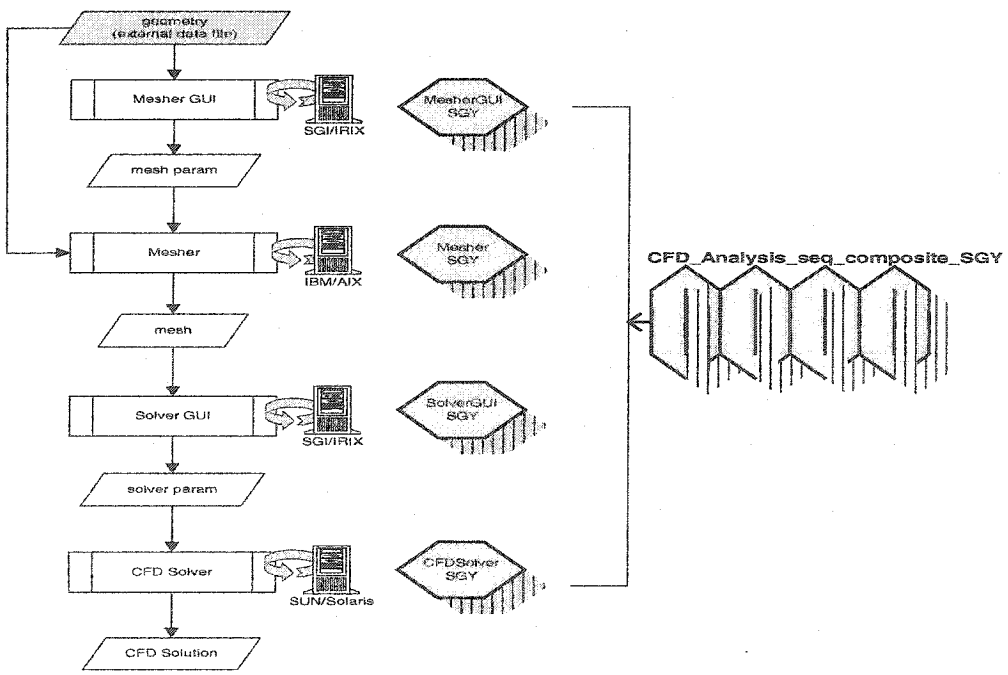


Appendix C: Flowcharts

C1 Creation of DataComponents Example



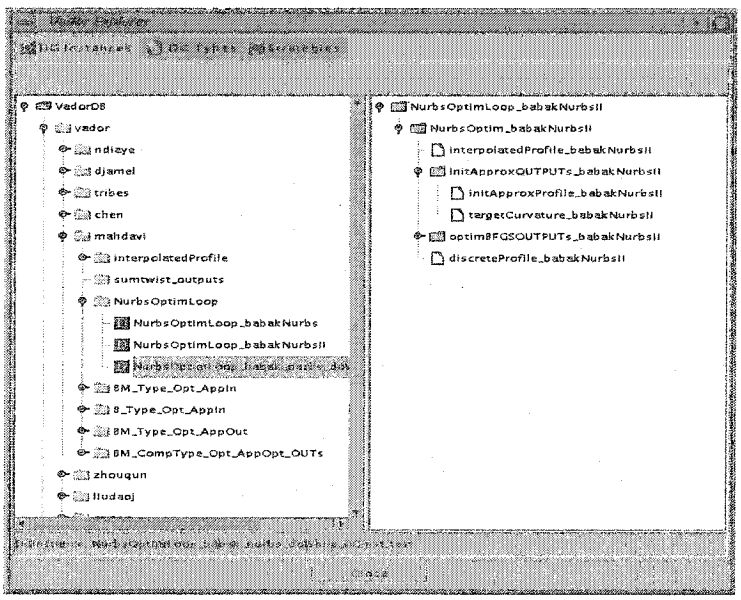
C2 Creation of StrategyComponents Example



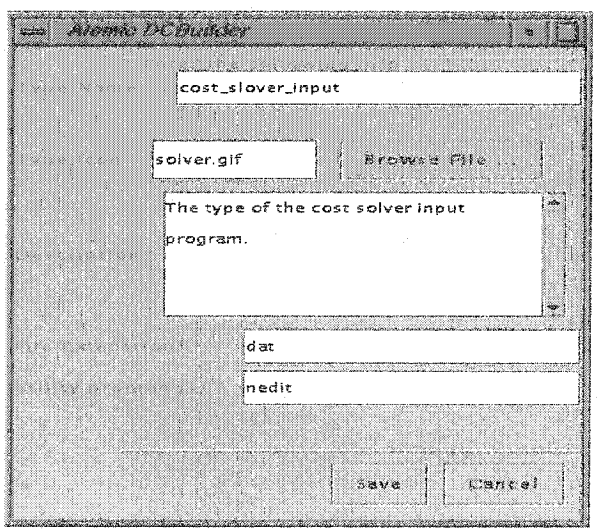
Appendix D: VADOR Images

More images from different parts of VADOR GUI, more specifically types and components browser as well as types and components builders can be found in this appendix.

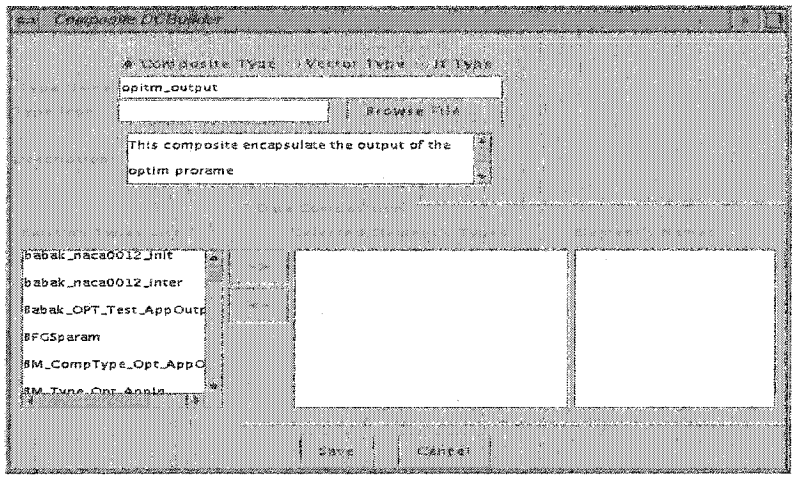
D1 VADOR Explorer



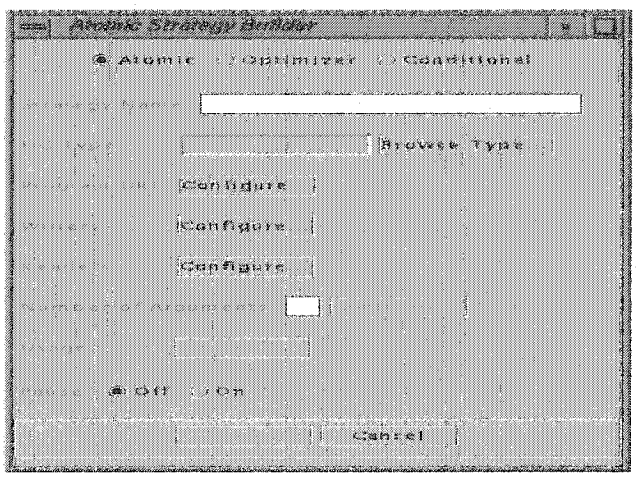
D2 Atomic DC Builder



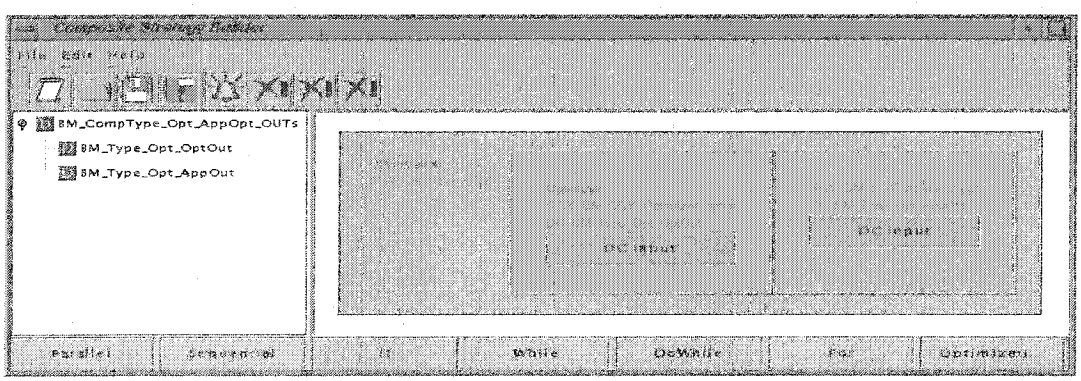
D3 Composite DC Builder



D4 Atomic StrategyComponent Builder



D5 Composite StrategyComponent Builder



References

[1] Renaud J. E., Aerospace and Mechanical Engineering, Notre Dame University, URL: http://www.nd.edu/~ame/facultystaff/Renaud.John.html

[2] Ontario Aerospace Council. Critical Technology Reports: Technical Report, 1996.

[3] Giunta, A. A., "Aircraft Multidisciplinary Design Optimization Using Design of Experiments Theory and Response Surface Modeling," Ph.D. Dissertation, Virginia Polytechnic Institute and State University, VA, 1997.

[4] Chen B., Liu, D., Mahdavi, B., Zhou, Q., Bouhemhem D., Ndiaye A., Guibault F., Ozell B., Pelletier, D., and Trépanier, J-Y., "A Data-centric Distributed Framework for Engineering Design," 48th Annual Conference of The Canadian Aeronautics and Space Institute, Toronto, ON, Canada, April 29 - May 2, 2001.

[5] Sobieszczanski-Sobieski, J., "Multidisciplinary Design Optimization MDO Methods: Their Synergy With Computer Technology in The Desing Process," The Aeronautical Journal, 1991.

[6] Ndiaye, A., Trépanier, J-Y., Guibault, F., Ozell, B., and Mahdavi, B., "Recommendations on Application Interfacing and Integration," Technical Report, CERCA, March, 2000.

[7] Ridlon, S. A., "A Software Framework for Enabling Multidisciplinary Analysis and Optimization," 6th AIAA/NASA/ISSMO Symposium on Multidiciplinary

Analysis and Optimization, Bellevue, WA, AIAA-96-4133-CP, pp.1280-1285, September, 1996.

[8] Townsend, J. C., Weston, R. P. and Eidson, T. M., "A Programming Environment for Distributed Complex Computing. An overview of the Framework for Interdisciplinary Design Optimization (FIDO) Project," Technical Report, NASA TM 109058, 1993.

[9] Weston, R. P., "FIDO - Framework for Interdisciplinary Design Optimization," 1996.

[10] Allwright, S., "Multidisciplinary Design, Analysis and Optimization of Aerospace Vehicles - the MDO Project," Royal Aeronautical Society MDO Conference, 1998.

[11] Ozell, B., Trépanier, J-Y., Mahdavi, B., Ndiaye, A., and Guibault, F., "Recommendations on Architectural and Technological Issues for the VADOR Framework," Technical Report, CERCA, June, 2000.

[12] Hulme, K. F., "The Design of a Simulation-Based Framework for the Development of Solution Approaches in Multidisciplinary Design Optimization," Ph.D. Dissertation, University of New York at Buffalo, 2000.

[13] Society of Concurrent Engineering (SOCE) / Society of Concurrent Product Development (SCPD), URL: http://www.soce.org/

[14] Concurrent Engineering Research Center (CERC), West Virginia University, URL: http://www.cerc.wvu.edu/

[15] Concurrent Engineering Research and Applications (CERA) Journal, Technomic Publishing Company, URL: http://www.ceraj.com/

[16] Concurrent Engineering Conferences (CECONF), URL: http://www.ceconf.com/

[17] Concurrent Engineering Team (CETEAM) / Concurrent Engineering Institute, URL: http://www.ceteam.com/

[18] Hartley, J. R., "Concurrent Engineering. Shortening Lead Times, Raising Quality, and Lowering Costs," Productivity Press, Cambridge, MA, USA, 1992.

[19] Carter, D. E. and Baker, B. S., "Concurrent Engineering: The Product Development Environment for the 1990's," Addison-Wesley Publishing Company, New York, NY, USA, 1991.

[20] Sobieszczanski-Sobieski, J., "A Linear Decomposition Method for Optimization Problems - Blueprint for Development," NASA Technical Memorandum 83248, 1982.

[21] Sobieszczanski-Sobieski, J., "Multidisciplinary Design Optimization: An Emerging, New Engineering Discipline," Advances in Structural Optimization, Kluwer Academic, 1995.

[22] Alzubbi, A., Ndiaye A., Mahdavi, B., Guibault, F., Ozell, B., and Trépanier, J-Y, "On the use of Java and RMI in the development of a computer framework for MDO," 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analaysis and Optimisation, Long Beach, California. AIAA 2000-4903, September 6-8, 2000.

[23] Sobieszczanski-Sobieski, J., and Haftka, R. T., "Multidisciplinary Aerospace Design and Optimization: Survey of Recent Developments," AIAA paper 96-

0711, 34th Aeospace Sciences Meeting and Exhibit, Reno, NV, January 1996.

[24] Rogers, J. L., Salas, A. O., and Weston, R. P., "A Web-Based Monitoring System for Multidisciplinary Design Projects," 7th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, September, 1998.

[25] AIAA Multidisciplinary Design Optimization Technical Committee (MDO TC),
URL: http://endo.sandia.gov/AIAA_MDOTC/faq/MDOTC_whatisMDO.html

[26] Ndiaye, A., Trépanier, J-Y, Guibault, F., Ozell, B., and Mahdavi, B., "Database Requirements for an MDO Software Framework," CFD2K, 8e Conférence annuelle de la Société canadienne de CFD, Montréal, Québec, Canada, Vol. 2, 721-728, June 11-13, 2000.

[27] Salas, A. O., and Townsend, J. C., "Framework Requirements for MDO Application Development," AIAA-98-4740, 1998.

[28] Ridlon, S. A., "A Software Framework for Enabling Multidisciplinary Analysis and Optimization," 6th AIAA/NASA.ISSMO Symposium on Multidiciplinary Analysis and Optimization, Bellevue, WA, AIAA-96-4133-CP, pp.1280-1285, September, 1996.

[29] Adaptive Modeling Language (AML), TechnoSoft, Inc., Ohio,
URL: http://www.technosoft.com/products.htm

[30] Zweber, J. V., Blair, M., Kamhawi, H., Bharatram, G., and Hartong, A., Structural and Manufacturing Analysis of a Wing Using the Adaptive

Modeling Language," 39th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Long Beach, CA, AIAA-98-1758-CP, pp. 483–490, April, 1998.

[31] Design Analysis Kit for Optimization and Terascale Applications (DAKOTA) Project, Sandia Nation Libratories, URL: http://endo.sandia.gov/DAKOTA/

[32] Eldred, M. S., Hart, W. E., Bohnhoff, W. J., Romero, V. J., Hutchinson, S. A., and Salinger, A. G., "Utilizing Object-Oriented Design to Build Advanced Optimization Strategies with Generic Implementation," 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, AIAA-96-4164-CP, pp. 1568–1582, September, 1996.

[33] Developmental Aeronautics Revolutionizing Wind-tunnels with Intelligent systems for Nasa (DARWIN), NASA Ames Research Center, URL: http://www.darwin.arc.nasa.gov

[34] Walton, J. D., Korsmeyer, D. J., Batra, R. K., and Levy, Y., "The DARWIN Workspace Environment for Remote Access to Aeronautics Data," AIAA 97-0667, January, 1997.

[35] Townsend, J. C., Weston R. P. and Eidson, T. M., "A Programming Environment for Distributed Complex Computing: an Overview of the Framework for Interdisciplinary Design Optimization (FIDO) project," Technical report, NASA TM 109058, 1993.

[36] Weston, R. P., Townsend, J. C., Eidson, T. M., and Gates, R. L., "A Distributed Computing Environment for Multidisciplinary Design," 5th

AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City Beach, FL, AIAA-94-4372-CP, pp. 1091–1097, September, 1994.

[37] Hale, M. A., Craig, J. I., Mistree, F., and Schrage, D. P., "DREAMS and IMAGE: A Model and Computer Implementation for Concurrent, Life-Cycle Design of Complex Systems," Concurrent Engineering: Research and Applications, Vol. 4, No. 2, pp. 171–186, June, 1996.

[38] Hale, M. A, and Craig, J. I., "Techniques for integrating computer programs into design architectures," 6th AIAA/NASA/ISSMO Symposium on Multidiciplinary Analysis and Optimization, Bellevue, WA, AIAA 96-4166-CP, pp.1594-1601, September, 1996.

[39] El Aichaoui, S., Hale, M., and Craig, J., "Building Design Applications Using Process Elements," 7th AIAA/USAF/NASA/ISSMO Symposium on Multidiciplinary Analysis and Optimization, St.Louis, MO, AIAA 98-4876, September, 1998.

[40] iSIGHT, Engineous Software, URL: http://www.engineous.com/

[41] Tong, S. S., Powell, D., and Goel, S., "Integration of Artificial Intelligence and Numerical Optimization Techniques for the Design of Complex Aerospace Systems," AIAA Paper 92-1189, February, 1992.

[42] Golovidov, O., Kodiyalam, S., Marineau, P., Wang, L., and Rohl, P., "Flexible Implementation of Approximation Concepts in an MDO Framework," 7th AIAA/USAF/NASA/ISSMO Symposium on Multidiciplinary Analysis and Optimization, St.Louis, MO, AIAA 98-4959, September, 1998.

[43] LMS Optimus, LMS International, URL: http://www.lmsintl.com/

[44] Guisset, P., and Tzannetakis, N., "Numerical Methods for Modeling and Optimization of Noise Emission Applications," ASME International Mechanical Engineering Congress and Exposition, Dallas,TX, 1997.

[45] Multi-DIsciplinary Computing Environment (MDICE), CFD Research Coporation, USA, URL : http://www.cfdrc.com/

[46] Multi-DIsciplinary Computing Environment for AEroelasticity (MDICE-AE), URL: http://www.va.afrl.af.mil/vaa/vaac/CAE/cfdrc.html

[47] George, J., Peterson, J., and Southard, S., "Multidisciplinary Integrated Design Assistant for Spacecraft (MIDAS)," 36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, New Orleans, LA, AIAA-95-1372-CP, pp.1790–1799, April 1995.

[48] Numerical Propulsion System Simulation (NPSS), NASA Glenn Research Center, Ohio, URL: http://hpcc.grc.nasa.gov/npssintro.shtml

[49] Evans, A., Lytle, J., Follen, G., and Lopez, I., "An Integrated Computing and Interdisciplinary Systems Approach to Aeropropulsion Simulation - NPSS," International Gas Turbine and Aeroengine Congress and Exhibition, Orlando, FL, June 1997.

[50] NPSS Industry Review, NASA Glenn Research Center, October 1999.

[51] Scott, A. T., "An Evaluation of Three Commercially Available Integrated Design Framework Packages for use in the Space Systems Design Lab,"

SSDL, School of Aerospace Engineering, Georgia Institue of Technology, April 2001.

[52] Phoenix Integration, URL: http://www.phoenix-int.com/

[53] Analysis Server and Model Center overviews, Phoenix Integration, URL: http://www.phoenix-int.com/publications/

[54] Authentication in Analysis Server and Model Center, Technical White Paper, Phoenix Intergration, August, 2001.

[55] Van der Valen, A., Kokan, D., Frommann, O., The Pointer MDO Framework, Synaps, 2000.

[56] Pointer / Epogy, Synaps Inc., URL: http://www.synaps-inc.com/software.html

[57] Becker, J. C., and Bloebaum, C. L., "Distributed Computing for Multidisciplinary Design Optimization using Java," 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, AIAA 96-4165-CP , pp. 1583–1593., September, 1996.

[58] Multidiciplinary Optimization and Design Engineering Laboratary (MODEL), Departement of Mechanical and Aerospace Engineering, University of New York at Buffalo, URL: http://www.eng.buffalo.edu/Research/MODEL/

[59] XCAT / CCAT, Extreme Lab, Indiana University, IN,

URL: http://www.extreme.indiana.edu/xcat/

URL: http://www.extreme.indiana.edu/ccat/

[60] Global Grid Forum, URL: http://www.gridforum.org/

[61] Information Power Grid (IPG), NASA, URL: http://www.ipg.nasa.gov/

[62] The Globus Project, URL: http://www.globus.org/

[63] Johnston, W. E., Vaziri, A., Tanner, L. A., Feiereisen, W. J., and Thigpen, W., "Information Power Grid," Detailed Document of the Approach and Goals for NASA's Information Power Grid, NASA Ames Resarch Center.

[64] Bouhemhem, D., Chen, B., Guibault, F., Liu, D., Mahdavi, B., Ndiaye, A., Ozell, B., Pelletier, D., Trépanier, J-Y., and Zhou, Q., "Software Requirements Document," Technical Report, CERCA, September, 2000.

[65] Common Object Request Broker Architecture (CORBA), Object Management Group (OMG), URL: http://www.omg.org/corba/

[66] Component Object Model (COM), Microsoft,

URL: http://www.microsoft.com/com/

[67] Distributed Component Object Model (DCOM), Microsoft,

URL: http://www.microsoft.com/com/tech/DCOM.asp

[68] JavaBeans, Sun Microsystems,

URL: http://java.sun.com/products/javabeans/

[69] Enterprise Java Bean (EJB), Sun Microsystems,

URL: http://java.sun.com/products/ejb/

[70] Bramley, R., Chiu, K., Diwan, S., Gannon, D., Govindaraju, M., Mukhi, N., Temko, B., and Yehuri, M., "A Component Based Service Architecture for Building Distributed Applications," Indiana University, IN.

[71] Unified Modeling Language (UML), Rational,

URL: http://www.rational.com/uml/

[72] What is Java Tecnology, Sun Microsystems,

URL: http://java.sun.com/java2/whatis/

[73] Berg, D., and Fritzinger, J. S., "Advanced Techniques for Java Developers," Wiley Computer Publishing, 1999.

[74] Java Applet, Sun Microsystems,

URL: http://java.sun.com/applets/

[75] Villacis, J., "A Note on the Use of Java in Scientific Computing," Indiana University, IN.

[76] Chen, B., Liu, D., Mahdavi, B., Zhou, Q., Bouhemhem, D., Ndiaye, A., Guibault, F., Ozell, B., Pelletier, D., and Trépanier, J-Y., "A Data-centric Distributed Framework for MDO Management," 6th International Conference on Computer Supported Cooperative Work in Design, London, ON, Proceeding, 279-284, July 12-14, 2001.

[77] Anouan, F., Bouchemhem, D., Chen, B., Guibault, F., Liu, D., Mahdavi, B., Ndiaye, A., Ozell, B., Trépanier, J-Y., and Zhou, Q., "VADOR User Guide," Technical Report, CERCA, February, 2002.

[78] Flanagan, D., "Java in a Nutshell," O'Reilly, 1999.

[79] The Desing Manager's Aid for Intelligent Decomposition (DeMAID),

URL: http://www.openchannelfoundation.org/projects/DEMAID/

[80] Rogers, J. L., "Reducing Desing Cycle Time and Cost Through Process Resequencing," International Conference on Engineering Design (ICED), 1997.

[81] Rogers, J. L., and McCulley, C. M., "Integrating a Genetic Algorithm into a Knowledge-based System for Ordering Complex Desing Porcesses," NASA-TM-110247.

[82] Salas, A. O., and Rogers, J. L., "A Web-Based System for Monitoring and Controling Multidiciplinary Desing Projects," NASA-TM-97-206287, 1997.

[83] Hulme, K. F., and Bloebaum, C. L., "Development of a Simulation-based Framework for Exploiting New Tools and Techniques in Multidisciplinary Design Optimization," ASMO UK/ISSMO Conference on Engineering Design Optimization, Ilkley, United Kingdom, pp.179-186, July, 1999.

[84] Kroo, I., "Computation-Based Design," White Paper, Aircraft Aerodynamics and Design Group, Stanford University, 1996.

[85] Sues, R. H., and Cesare, M. A., "An Innovative Framework for Reliability-Based MDO," AIAA-2000-1509, 2000.

[86] Sues, R. H, Aminpour, M. A., Shin, Y., "Reliability Based MDO for Aerospace Systems," AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, 42nd, Seattle, WA, April 16-19, 2001.

[87] Probabilistic Finite Element System (ProFES), URL: http://www.profes.com/

[88] Cesare, M. A., and Sues, R. H., "PorFES Probabilistic Finit Element System--Bringing Probablistic Mechanics to the Destop" AIAA 99-1607, 1999.

[89] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., "Desing Pattern – Elements of Reusable Object-Oriented Software," Addison-Wesley, 1998.

[90] Sankar, M. R., Isaacs, A., Mujumdar, P. M., and Sudhakar, K., "MDO Framework Development - A Case Study With An Elementary Model Of Airborne Early Warning System Optimization."

[91] Sistla, R., Dovi, G., Su, P., and Shan, R., "Aircraft Desing Problem Implementation under the Common Request Broker Architechtrue

(CORBA)," 40th AIAA/ASME/ASCE/AHS/ASC SDM Conference, MO, April 1999.

[92] Barth, T., Grauer, M., Freisleben, B., and Thilo, F., "Distributed Solution of Simulation-Based Optimization Problems on Networks of Workstations," University of Siegen, Germany.

[93] Zhou, Q., Mahdavi, B., Liu, D., Guibault, F., Ozell, B., and Trépanier, J-Y., "A Web-based Distribution Protocol for Large Scale Analysis and Optimization Applications," 15th Annual International Symposium on High Performance Computing Systems and Applications, Windsor, Ontario, Canada, June 18-20, 2001.

[94] Cao, J., and Bennett, G. K. Z., "Direct Execution Simulation of Load Balacing Algorithm with Real Workload Distribution," The Journal of System and Software, p.p. 227-237, 2000.

[95] Wolffe, G. S., Hosseini, S. H., and Vairavan, K., "An Experimental Study of Workload Indices for Non-dedicated, Heterogenous System," University of Wiscousin.

[96] Kalyani, M. A. L., Wait, R., and Ranasighe, D. N., "Load Balacing Techniques for Distributed Memory Multiprocessor," January, 2001.

[97] Vanderplaats, G. N., "Numerical Optimization Techniques for Engineering Design: With Applications," McGraw Hill, New York, 1984.

[98] MySQL database, MySQL AB, URL: http://www.mysql.com

[99] Bouhemhem, D., Chen, B., Guibault, F., Liu, D., Mahdavi, B., Ndiaye, A., Ozell, B., Pelletier, D., Trépanier, J-Y., and Zhou, Q., "Damage Tolerance Analysis Interfacing, Automation and Integration," Technical Report, CERCA, September, 2000.

[100] Extensible Markup Language (XML), W3C, URL: http://www.w3.org/XML/

[101] CRAY supercomputers, URL: http://www.cray.com/products/systems/

[102] Geist, G. A., Kohl, J. A., and Papadopoulos, P. M., "PVM and MPI: a Comparison of Features," Calculateurs Paralleles Vol. 8 No. 2, pp. 137-150 June, 1996.

[103] Eldred, S. M., Giunta, A. A., Bloemen Waanders, G. B., Wojtkiewicz, F. S., Hart, E. W., and Alleva, P. M., "DAKOTA, a Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantificiation, and Sensitivtiy Analaysis," SAND Report, SAND2001-3796, Sandia National Laboratoiries, April, 2002.

[104] Schreiner, A. J., Trosin, J. P., Pochel, A. C., and Koga, J. D., "DARWIN – Intergrated Instumentaion and Intelligent Daabase Elements," NASA Ames Research Center.

[105] Fukunaga, A., and Stechert, D. A., "An Evolutionary Optimization System for Spacecraft Design," Proceedings of Tengh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Atlanta, GA, 1997.

[106] Fukunaga, A., Chien, S., Mutz, D., Sherwood, R., and Stechert, A., "Automating the Process of Optimization in Spacecraft Design," Proceedings of the 1997 IEEE Aerospace Conference, Snowmass, CO, pp.411-428, vol 4, 1997.

[107] Fukunaga, A., Stechert, A., and Chien, S., "Towards a Self-Configuring Optimization System for Spacecraft Design," Proceedings of International Symposium on Artificial Intelligence, Robotics and Automations in Space, Tokyo, Japan, 1997.

[108] Kale, V., "Implementing SAP R/3: The Guide for Business and Technology Managers," SAM, 2000.

[109] Systems, Applications, Product (SAP), URL: http://www.sap.com/

[110] BaaN, URL: http://www.baan.com/

[111] Oracle, URL: http://www.oracle.com/

[112] PeopleSoft, URL: http://www.peoplesoft.com/

[113] J.D. Edwards, URL: http://www.jdedwards.com/