다분야통합최적설계를 위한 지능형 분산 시스템

이 재호* • 홍은지**

A Distributed Intelligent System for Multidisciplinary Design Optimization

Jaeho Lee • Eunji Hong

요 약

산업 및 가정용 기기들이 점차 복잡해짐에 따라 다양한 공학 분야의 해석 기술을 동시에 고려하면서 이들 원리를 적용한 최적의 설계를 결정하는 방법론의 필요성이 대두되고 있다. 다분야통합최적설계 또는 MDO(Multidisciplinary Design Optimization)라 일컫는 새로운 기술은 이러한 필요에 대응하는 기술로서 국내외적으로 활발한 연구가 진행되고 있다. 이러한 MDO 기술을 구현하는 소프트웨어와 하드웨어 복합체계를 MDO 프레임웍(framework)이라 한다. 일반적으로 프레임웍이란 실제 응용프로그램의 용도에 맞는 주문제작(customization)이 가능한 일종의 전단계 프로그램이라 할 수 있다. MDO 프레임웍은 설계 및 해석 도구들간의 인터페이스를 제공하고, 이들 도구들이 사용하는 설계 데이터를 효율적으로 공유할 수 있도록 지원하여, 설계 작업을 정의, 실행, 관리하는 역할을 한다. 이러한 MDO 프레임웍은 설계 작업을 통합적으로 관리하고 자동화하여 설계 도구간의 데이터 전달과 변환에 소요되는 설계자의 부담을 경감시키며 다분야 전문가가 참여하는 공통 작업 환경을 제공함으로써 설계 효율성을 증진시킨다. 본 논문에서는 이러한 효용을 달성하기 위한 MDO 프레임웍(framework)을 제시하고 프레임웍 설계의 논리적 근거와 타당성을 밝힌다. 본 논문에서 제안하는 다분야 통합 최적화를 위한 분산형 지능 시스템인 DisMDO는 사용자가 GUI를 통해서 편리하게 다분야통합최적화 문제를 해결할 수 있도록 지원하며, 제공되는 스크립트 언어를 통해서도 이를 정의할 수 있도록 지원하여 일괄처리도 가능하도록 한다. 또한, 집중화된 데이터베이스를 관리하여 다분야 전문가들이 공통의 데이터를 안전하게 공유할 수 있도록 지원하며, 외부에서 제공되는 해석 도구나 최적화 모듈을 손쉽게 프레임웍에 통합시킬 수 있도록 하는 인터페이스 제작기 (factory) 기능을 제공한다.

1. 서론

제품설계에 관련된 다양한 공학해석분야, 즉 구조해석, 동역학, 열 유체 유동해석, 제어, 전자기장 해석 등을 동시에 고려하면서 최적의 설계를 결정하는 것을 다분야통합최적설계 (Multidisciplinary Design Optimization : MDO) 기술이라 한다. 점차 복잡해 지는 공학 시스템에 대한 수요 증가로 이러한 MDO 기술 개발이 공학 설계의 필수적인 요소로 대두되고 있다.

전통적으로 각 공학해석분야의 원리들은 독자적인 컴퓨터 시스템으로 구현되어 있다. 이러한 시스템들은 상호 연관성을 고려하지 않고 개발되었기

* 서울시립대학교 전기전자공학부
** 성공회대학교 컴퓨터정보학부

때문에, 여러 가지 원리가 적용되어야 하는 복잡한 문제를 해결하기 위해서는 설계자가 이들 시스템들을 순차적으로 적용하면서, 각 시스템의 결과를 다른 시스템에 적합한 형태로 다시 가공해야 하는 소모적 과정을 거쳐야 한다. 그 동안 각 분야별 해석 기술이 많이 발전하여 그 난이도와 정밀도는 점차 높아져 왔으나 이에 따른 설계의 복잡성도 증대되어 분야간 협동설계가 더욱 어렵게 되었으며, 통합적인 설계의 필요성은 점증하고 있다.

대부분의 국내 산업체 설계 관행을 보면, 각 설계부서에서 작업한 결과간의 상충이 생기면 관리자가 판단하거나 또는 부서간 협의에 의하여 재설계하는 과정을 반복한다. 따라서 최종 결과가 시스템 수준의 최적설계이기를 기대하기는 어려운 실정이며 설계 시간도 상당히 소요되므로 체계적인 최적화 방법의 적용이 필요하다. 다분야통합최적설계 기술은 다양한 분야의 공학적 설계 원리들을 동시에 고려하여, 균형 있고 유기적인 방법으로 최적의 설계를 결정하는 체계적인 설계자동화 기술이다. 따라서, MDO 기술을 개발하여 산업체의 설계현장에 이전하면, 산업체는 경쟁력 있는 제품을 짧은 시간에, 적은 비용으로 설계할 수 있는 체계를 갖출 수 있게 된다.

미국 산업체의 경우 GM과 Ford 등의 자동차회사, 보잉 등의 항공우주업체들이 MDO 기술에 대한 연구를 활발히 진행하고 있고, 점차 조선, 화공, 반도체 분야 등 전 공학분야로 기술의 적용범위가 확산되어 나가는 추세이다.

전형적으로, 공학자는 설계 과정에서 수 많은 프로그램들을 사용한다. 예를 들어, CAD(Computer Aided Design)를 통해 모형을 설계한 후 각 원리들을 해석 도구 (analysis tool) 또는 CAE(Computer Aided Engineering)를 통해 시뮬레이션(simulation)하고, 설계를 최적화 하기 위해 최적화(optimization) 모듈을 사용한다. 여기에 여러 원리들 사이에 일어날 수 있는 상호작용(interaction)까지 고려한 MDO 문제를 해결하기 위해서는 컴퓨터 소프트웨어 기술과 정보기술, 다분야통합최적설계기술 및 기존 CAD/CAE 도구를 통합하여 MDO 프레임웍(framework)을 구축해야 한다. MDO 프레임웍은 설계자가 최소의 프로그래밍 노력으로 양질의 설계를 생산할 수 있도록 하여 시간과 경비를 절약할 수 있게 해준다[Salas 98]. MDO 프레임웍(Framework)이란 MDO 기술이 구현된 소프트웨어와 하드웨어 복합체계를 일컫는 것으로, 재사용이 가능한 반 완성(semi-completed) 응용 프로그램이다. 일반적으로 프레임웍이란 실제 응용프로그램의 용도에 맞는 주문제작(customization)이 가능한 일종의 전단계 프로그램이라 할 수 있다. MDO 프레임웍은 MDO 응용의 다양한 처리(process)를 통합(integration)하는 기능을 제공하므로, 설계자는 해당 응용 분야에만 더욱 집중할 수 있다. 또한, MDO 프레임웍은 MDO 응용의 공통적인 작업 환경(environment)을 제공하므로, 다분야최적설계 프로젝트의 생산성을 향상시킬 수 있다.

기존에 나와 있는 Optimus, IMAGE, iSIGHT와 같은 최적설계를 위한 프레임웍은 다분야통합최적설계를 위한 환경으로는 부족한 점이 많다.

본 논문에서는 MDO에 대해 살펴보고, 프레임웍이 만족해야 할 요구 사항에 대해 알아 본 후, 기존의 시스템들의 문제점을 살펴본다. 이러한 분석을 바탕으로 새로운 MDO 프레임웍인 DisMDO (Distributed Intelligent System for MultiDisciplinary Optimization)를 제안한다. DisMDO는 사용자가 GUI(Graphical User Interface)를 통해서 편리하게 다분야통합최적화 문제를 해결할 수 있도록 지원하며, 제공되는 스크립트 언어를 통해서도 이를 정의할 수 있도록 지원하여 일괄처리도 가능하도록 한다. 또한, 집중화된 데이터베이스를 관리하여 다분야 전문가들이 공통의 데이터를 안전하게 공유할 수 있도록 지원하며, 외부에서 제공되는 해석 도구나 최적화 모듈을 손쉽게 프레임웍에 통합시킬 수 있도록 하는 인터페이스 제조기(factory) 기능을 제공한다.

2. 다분야통합최적설계 (MDO)

공학 분야에서 일컫는 최적화 설계란 목적 함수

(objective function)라고 부르는 함수의 최소값 또는 최대값을 구하는 과정이다. 이 과정에 의한 결과로써 설계변수가 만족스럽기 위해서는 설계 값에서 구속조건(constraint)이라고 일컫는 등식 또는 부등식들을 만족해야만 한다. 즉 최적설계라는 과정은 구속조건을 만족하면서 목적함수의 최소 또는 최대값을 구하는 것이다.

MDO 문제는 복잡하게 결합된(coupled) 공학 시스템을 기술하는(describe) 최적화 문제(optimization problem)이다[Alex 98].

3. MDO 구성요소

본 절에서는 MDO를 구성하는 개념적인 구성요소에 대해 살펴 본다[Sobi 96].

3.1. 시스템의 수학적 모델링

공학 시스템에서는 각 공학분야원리를 수학적 모델로 표현하고, 이러한 수학적 모델을 소프트웨어 모듈(module)로 구현한다. 각 모듈은 물리적 현상, 물리적 부분, 또는 시스템의 또 다른 부분을 표현한다. 모듈사이의 데이터 전달(transfer)은 시스템 내부의 결합(coupling)에 의해 결정된다. 이러한 결합에서 전달되는 데이터의 양은 입출력 비용으로 직결된다. 따라서, 이러한 데이터 전달 양을 절감하고, 효율적으로 데이터를 전달할 수 있게 하는 방법이 필요하다.

또한 계산 비용(computational cost)의 중요성도 점점 강조되는 경향이므로, MDO는 정확도(accuracy)와 비용 사이의 상충관계(trade-off)를 고려해야 한다. 따라서, MDO에서 사용하는 모델들은 보통 단일 설계 최적화 (single disciplinary optimization) 때 보다 보통 덜 복잡하고 덜 정확한 모델을 사용한다.

3.2. 설계-지향 분석

설계자들은 자신이 설계한 결과가 어떤 식으로 표현될 지를 빠른 시간 내에 시뮬레이션 해 보기를 원한다. 이러한 요구를 만족하기 위해서 시뮬레이션을 행하는 해석 도구(analysis tool)는 다음과 같은 특성을 지녀야 한다.

우선, 저 비용에서부터 정확한 결과를 내는 고비용 분석까지 여러 수준 중에 하나를 선택할 수 있어야 한다. 설계 변경에 의해 영향 받은 부분만 다시 분석할 수 있는 지능이 있어야 하고, 입력(input)에 대응되는 출력(output)의 민감도 미분계수(sensitivity derivatives)를 계산할 수 있어야 한다. 또한, 설계 과정에서 전형적으로 생성(generate)되는 거대한 양의 데이터를 관리하고 시각화(visualization) 할 수 있는 하부구조(infrastructure)가 제공되어야 한다.

3.3. 근사 개념

설계 변수의 개수가 많아지면, 목적 함수와 구속 조건을 계산하는 비용이 너무 높아져서, 정확한 MDO 분석으로 수행하기 힘든 경우가 있다. 또한 어떤 원리를 구현한 해석 도구는 노이즈(noise)를 포함하는 다듬어지지 않은 반응(response)을 만들어 낼 수 있다. 이와 같은 이유로 대부분의 복잡한 공학 시스템에서는 목적 함수와 구속 조건을 근사화(approximation)하여 사용한다

3.4. 시스템 민감도 분석

민감도해석은 최적설계의 결과 값에서 문제의 여러 변수들에 대한 최적설계의 변화를 표현한다. 즉 최적 설계 값에서 어떤 변수가 약간 변화하였을 때, 최적 설계 값이 얼마나 변화하는 지를 해석하는 것이다. 보다 포괄적인 의미에서는 목적함수나 구속조건의 설계변수에 대한 변화도 포함한다고 볼 수 있다. 이러한 민감도해석은 그 자체만으로도 의미가 있으며, 최적화기법 또는 MDO 방법론에 따라 필수적으로 포함되어야 하는 구성요소이기도 하다. 민감도해석 방법에는 유한차분법과 같은 수치적인 방법과 최적화조건을 이용하는 해석적인 방법들이 있다.

4. MDO 방법론 및 예제

MDO 문제를 해결하기 위해서 여러 가지 방법론들이 대두되었다. MDF (Multidisciplinary Feasible Method), IDF (Individual Discipline Feasible Method), CO(Collaborative

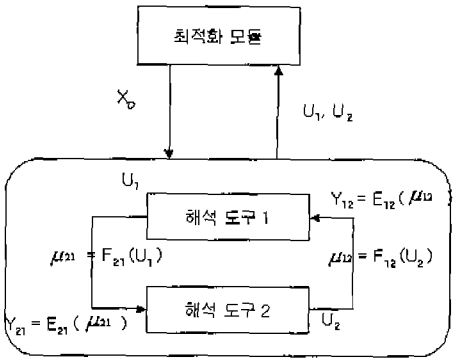
Optimization) 등이 대표적인 방법론이다. 본 논문에서는 이러한 방법론에 대해 모두 설명할 필요는 없다고 판단되나, 대표적인 MDF 방법론을 설명하고, 예제를 살펴보는 것은 의미 있는 일이라 여겨진다.

MDF는 MDO 문제 해결을 위해 사용되는 해결 방법 중에 가장 널리 알려져 있는 접근 방식이다. 여기에서는 설계 변수 벡터 X_D가 해석기들이 결합된(coupled) 시스템에 제공된다. 제공받은 X_D 값을 완전한 다중통합 해석도구(multidisciplinary analysis)가 고정점 반복(fixed point iteration)을 수행한다. 이렇게 반복하는 이유는 목적(objective) 함수 $F(X_D, U(X_D))$와 구속조건(constraint) $g(X_D, U(X_D))$을 계산하여 평가하는데 사용되는 시스템 (MDA) 출력 변수 $U(X_D)$를 얻기 위해서 이다. 최적화 문제는 다음과 같이 정의된다.

Minimize: $F(X_D, U(X_D))$

Subject to: $g(X_D, U(X_D)) \leq 0$

and bounds on design variables, X_D



[그림 1] MDF 모델

[그림 1]은 MDF 해석과 최적화에서의 데이터 흐름을 보여준다. 이 그림에서, μ_{ij}는 해석 도구 j의 출력에 적당한(fit) F_{ij}를 사용하여 얻은 스플라인 계수(spline coefficient)이다. F_{ij}는 보간법 (interpolation) 또는 근사법(approximation)에 의해 얻은 값일 수 있다. 사상(mapping) E_{ij}는 해석 도구 j에서 나오는 스플라인 표현 (representation)을 해석 도구 i에서 사용되기 적합한 형태로 표현해 주기 위해서 사용된다.

이제, MDO 문제 예를 살펴본다. 전자 패키징 (electronic packaging)은 전기 서브시스템 (electrical subsystem)과 열 서브시스템(thermal subsystem)이 결합된 다분야 통합 문제 (multidisciplinary problem)이다. 저항(resistance)은 온도를 조정함에 따라 영향을 받고, 온도는 저항에 따라 결정된다.

이 문제의 목적 함수는 구속조건을 만족하면서 전자 패키징을 위한 와트 밀도(watt density)를 최대화하는 것이다. 저항기(resistor)를 위한 온도는 특정 온도 이하이어야 하고, 두 저항기에 흐르는 전류(current)는 같아야 한다는 구속 조건이 있다.

MDF 방식에 따르면, 최적화 문제는 다음과 같이 주어진다.

Maximize: Y_1 (Watt Density)

Subject to:

$h_1 = Y_2 - Y_3 = 0.0$ (branch current equality)

$g_1 = Y_{11} - 85.5 \leq 0$ (component 1 reliability)

$g_2 = Y_{12} - 85.5 \leq 0$ (component 2 reliability)

위와 같은 MDF 문제는 다음과 같은 8 개의 설계 변수를 가진다.

0.05 ≤ 열 씽크(sink) 폭(width) (x_1) ≤ 0.15

0.05 ≤ 열 씽크(sink) 길이(length) (x_2) ≤ 0.05

0.01 ≤ 핀(fin) 길이 (x_3) ≤ 0.10

0.005 ≤ 핀(fin) 폭 (x_4) ≤ 0.05

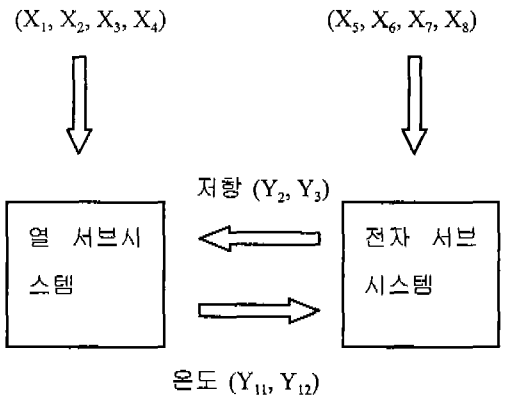
10.0 ≤ 저항(resistance) #1 (x_5) ≤ 10000.0

0.004 ≤ 온도 계수 (x_6) ≤ 0.009

10.0 ≤ 저항(resistance) #2 (x_7) ≤ 10000.0

0.004 ≤ 온도 계수 (x_8) ≤ 0.009

해석 도구 사이의 상호작용을 그림으로 나타내면 [그림 2]와 같다.



[그림 2] 해석 도구 사이의 상호 작용

5. MDO 프레임웍

이러한 MDO 문제를 해결하기 위해서는 컴퓨터 소프트웨어 기술과 정보기술, 다분야 통합 최적설계기술 및 기존 CAD/CAE 도구를 통합하여 MDO 프레임웍을 구축해야 한다. MDO 프레임웍의 목적은 MDO 문제 해결을 위한 응용을 개발하고 실행하기 위함이다.

5.1. MDO 프레임웍을 이용한 MDO 문제 해결 과정

본 절에서는 MDO 프레임웍을 이용하여 사용자가 MDO 문제를 해결하는 과정을 간단하게 살펴보고, MDO 프레임웍이 만족해야 할 요구사항을 정리한다[Krish 98, Salas 98].

5.1.1. 문제 정의 단계

제일 먼저 하는 작업은 MDO 문제를 정의하는 것이다. 어떤 MDO 문제를 해결해야 하는지를 결정하고, 목적 함수와 구속조건 들을 정의하며, 어떤 설계 변수들을 정의해야 하는지를 결정한다. 각 설계 변수들의 하한, 상한 값을 정의하고, 어떤 해석 도구와 최적화 모듈을 사용할 지를 결정한다.

5.1.2. 프로세스 구성 단계

최적화 문제를 정의하고 나면, 처리 순서를 결정한다. 프로세스 구성은 최적화 문제를 정의하는 과정에서 결정될 수도 있다. 프로세스 구성은 어떤 MDO 방법론을 사용할 지에 따라 달라질 수 있다. 분기, 반복 조건 및 횟수, 데이터 흐름, 제어 흐름 등이 표현된다. 여기까지 정의하고 나면, MDO 문제를 위한 MDO 응용이 정의되었다고 할 수 있다.

5.1.3. 실행 및 감시 단계

문제 정의와 프로세스 구성이 완료되고 나면, 사용자가 정의한 대로 MDO 응용을 실행한다. MDO 응용을 실행하는 동안, 이력(history) 정보들을 감시할 수도 있고, 일시 중단한 후 다시 시작할 수도 있다.

5.1.4. 후처리 단계

실행 단계가 완료되거나 실행을 일시 중지한 후에 실행과정에서 발생하여 저장된 데이터를 이용하여 분석작업을 수행할 수 있다. 차트나 그래프를 이용한 데이터 분석 자료의 제시도 이루어 지며 다른 분석 프로그램에서 데이터베이스에 저장된 데이터를 체계적으로 접근할 수 있는 응용 프로그램 인터페이스를 제공할 수 도 있다.

5.2. MDO 프레임웍의 요구 사항

MDO 프레임웍이 갖추어야 할 요구사항을 시스템 구조, 문제 정의, 실행 관점에서 알아본다.

5.2.1. 시스템 구조

- **툴 박스 기능**

과거(legacy) 또는 최근(modern)에 개발된 공학 해석 도구, 최적화 모듈을 사용할 수 있게 하는 툴 박스 기능이 제공되어야 한다. 프레임웍은 확장 가능해야 하며, 시스템에 새로운 프로세스를 통합하기 위한 인터페이스를 개발할 수 있도록 지원해야 한다.

- **객체 지향 개념**

프레임웍에는 객체지향 개념이 적용되어야 한다. 객체지향 개념은 확장성과 캡슐화, 오버라이딩 등을 제공하여, 툴 박스 기능 추가 및 분산 환경 제공을 용이하게 한다.

- **높은 성능**

최적화 과정에 너무 많은 오버헤드(overhead)를 가지게 해서는 안된다. 사용자가 최적화를 위한 코드를 잘 정제하지 못하면 속도 저하가 발생할 수 있게 되는데, MDO 프레임웍은 현재 어디에서 시간을 지체하는 활동이 발생하고 있는지를 파악할 수 있도록 성능을 측정할 수 있는 방안을 제공해야 한

다.

◆ 큰 문제 처리

MDO 프레임웍은 커다란 문제도 처리할 수 있어야 한다. 현재는 보통 수백 개 정도의 설계 변수를 가지는 문제를 처리하고 있는데, 미래에는 수천 개의 설계 변수를 가지는 문제도 처리할 수 있어야 한다.

◆ 협동 설계

하나의 MDO 문제를 해결하는 데는 보통 여러 분야의 전문가들이 포함되므로, 이들이 협동해서 같이 문제를 해결할 수 있도록 하는 환경이 필요하다. MDO 프레임웍은 문제 해결을 위한 데이터에 여러 사용자가 동시 접근할 수 있는 시스템 구조를 제공해야 한다.

◆ 표준

MDO 프레임웍은 메시지 전달, 분산 환경, 데이터베이스 접근, 언어 등에 있어서 국제적인 표준(standard)에 기반을 두어야 한다.

5.2.2. 문제 정의

◆ GUI 및 제어 구조

사용자가 프로그래밍을 하지 않고서도 쉽게 복잡한 분기 및 반복 조건 등을 표현할 수 있도록 지원해야 한다. 프로세스를 연결할 수 있도록 시각적 프로그래밍 인터페이스(visual programming interface)를 제공하는 것이 바람직하다.

◆ 기존 코드와의 통합

여러 가지 언어로 작성된 기존 코드나 소스 코드(source code)가 제공되지 않는 모듈 등을 시스템에 통합하여 사용할 수 있어야 한다. MDO 프레임웍은 적당한 입력 파일을 생성할 수 있고 해석 프로그램을 호출하고 관심 있는 출력 값을 자동적으로 추출할 수 있는 래퍼(wrapper)를 생성하는 툴을 제공해야 한다.

◆ 다중 수준 문제 정의

MDO 프레임웍은 사용자가 여러 가지 해석 도구와 여러 최적화 방법을 통합하여 부문제(subproblem) 정의도 가능하도록 지원해야 한다. 모든 문제에 가장 좋은 하나의 최적화 방법이 존재하지 않으므로, 서로 다른 방법론을 적용해 볼 수 있도록 하는 것이 중요하다. 사용자가 해당 MDO 문제에 맞는 최적화 방법론을 선택할 수 있도록 해야 한다.

◆ MDO 문제 재구성

이미 해결했던 MDO 문제를 이용하여 유사한 MDO 문제 정의를 재 구성할 수 있도록 지원해야 한다.

◆ 프로세스 수행 및 데이터 이동

MDO 프레임웍은 프로세스 수행과 데이터 이동을 자동화 시킬 수 있어야 한다. 전통적인 다분야 통합 설계 방법에서는, 공학자는 하나의 해석기에서부터 받은 데이터를 기다려서 그것을 재구성한 후 다시 다른 해석기에 입력으로 전달해야 했다. MDO 프레임웍은 입력 파일의 준비나 해석 도구 또는 최적화 모듈의 수행, 출력 파일에서의 적당한 데이터 추출, 프로세스 간의 데이터 이동 등을 자동화하여 이러한 지연이 발생하지 않도록 한다.

5.2.3. 실행

◆ 병렬 수행

여러 해석 도구가 사용되는 MDO 문제의 경우, 어떤 해석 도구들은 병렬 수행을 해도 서로 간섭하지 않고 정확한 값을 생산해 낼 수 있다. 이러한 해석 도구들을 병렬 수행시켜 성능을 향상시킬 수 있다.

◆ 이질 분산 환경

MDO 프레임웍은 해석 도구나 필요한 코드들이 이질 분산 환경에 흩어져 있는 경우에도 작동할 수 있도록 지원해야 한다.

◆ 사용자 상호작용

MDO 프레임웍은 사용자가 정의한 MDO 문제를 수행해서 제대로 결과가 나오나 확인해 보고, 해석 도구를 교체해보고, 제어 구조를 변경하는 등

의 작업을 통해 MDO 문제 풀이를 정제해 나가는 작업을 수행할 수 있도록 환경을 제공해야 한다.

• **일괄 처리 모드**

사용자가 GUI를 통해 상호작용을 하지 않고 일괄 처리(batch) 모드로 문제를 해결할 수 있도록 하는 방법도 제공해야 한다.

• **데이터베이스**

대형 문제를 해결하기 위해서는 MDO 문제를 위해 사용되는 데이터를 집중화 된 데이터베이스로 관리하는 것이 권장된다.

• **중간 결과 및 최종 결과 디스플레이**

사용자가 선택한 설계 변수나 구속조건, 목적 함수의 값들이 어떻게 변화되는 지를 쉽게 볼 수 있는 이력(history) 정보들이 보기 편하게 시각화(visualization)되어 디스플레이 되어야 한다.

• **모니터링**

현재 수행 상태를 모니터(monitor)할 수 있는 기능을 제공해야 한다.

• **결함 내성**

어떤 프로세스가 실행에 실패했을 경우, 데이터의 손실 없이 이전 체크포인트 시점까지 되돌아갈 수 있다던가, 실행을 잠시 중단했다가 다시 시작시킬 수 있는 기능이 제공되어야 한다.

5.3. 기존 시스템

위와 같은 요구 사항을 모두 만족하는 MDO 프레임웍은 아직 없다. 본 절에서는 기존에 존재하는 프레임웍들의 특징 및 장단점을 살펴본다.

5.3.1. iSight

iSight[iSight 99]는 Engineous Software에서 상업적으로 개발된 프레임웍이다. iSight는 외부에서 개발된 해석 코드, 모니터링 툴 등을 통합하여 사용할 수 있게 해 준다. 해석 도구 사이의 데이터 교환을 가능하게 하며, 여러 가지 최적화 모듈이 사용될 수 있도록 지원한다. 하지만, 이질 분산 환경에서의 수행이 힘들며, 협동 작업을 위한 지원이 없고, 집중화 된 데이터베이스 기능을 제공하지 않는다. 또한, GUI를 통해 문제 정의 및 프로세스 구성을 할 수 있지만, 간단한 문제에 대해서만 가능하고, 복잡한 MDO 문제를 정의하기 위해서는 iSight에서 제공하는 MDOL 이라는 스크립트 언어(script language)를 이용하여 프로그래밍해야 하는 번거로움이 있다.

5.3.2. LMS Optimus

Optimus는 LMS Numerical Technologies에서 상업적으로 개발된 프레임웍이다. LMS Optimus는 사용자가 문제를 정의하고 사용할 방법론을 선택하고 결과를 분석할 수 있게 해 준다. 입력 파일/출력 파일 형태로 설계 변수의 입력 출력을 정의할 수 있는 해석 도구들을 통합하여 사용할 수 있다. 여러 종류의 최적화 모듈을 자체적으로 제공하며, 외부에서 개발된 최적화 모듈을 포함시킬 수도 있다. 하지만, 최대 설계 변수의 개수가 50개로 제한되며, iSight와 마찬가지로 복잡한 제어 구조를 가지는 MDO 문제를 정의하기 위해서는 명령어 파일(command file)을 직접 조작해야 한다,

5.3.3. IMAGE

IMAGE[IMAGE 97]은 Georgia Institute of Technology에서 개발된 연구 프로젝트이다. IMAGE는 객체지향 데이터 관리 기능을 제공하며, 분산 처리 기능을 제공한다. IMAGE는 다분야 해석 도구를 위한 프레임웍을 제공하지만, 다분야통합최적화를 위한 프레임웍은 아니다. 또한, 집중화 된 데이터베이스를 제공하지 않는다.

6. DisMDO의 설계와 구현

본 절에서는 앞서 제시한 요구사항 분석을 토대로 설계한 우리의 DisMDO 프레임웍을 설명한다. 특히 요구사항을 반영하기위해 채용된 기술과 논리적 근거를 제시한다.

6.1. MDO 관련자 분류

DisMDO 프레임웍은 설계작업에 관여하는 다양한 사용자를 위한 사용자별 관점(view)과 역할 및 기능을 고려하여 설계되었다. 이러한 MDO 프레임웍의 사용자들은 UML 기반 소프트웨어 개발에 사

용되는 쓰임새(use case)의 행위자(actor) 역할을 한다.

6.1.1. 최종사용자 (End User)

DisMDO의 최종사용자는 DisMDO에 이미 구현되어있는 MDO 방법론을 이용해서 문제를 정의하고 해결한다. 숙련된 최종사용자는 필요와 숙련도에 따라 process configuration과 post processing 작업에도 관여 할 수 있다. 이러한 사용자는 해결하고자 하는 문제에 관해서는 잘 알고 있으나 다양한 MDO 방법론에 대해서는 자세히 알지 못해도 프레임웍을 사용할 수 있어야 한다. 따라서 DisMDO는 최종사용자가 해결해야 할 문제를 구성하고 실행하여 구성된 MDO 문제를 해결하는 역할을 수행한다. 이는 단순 MDO 응용자에게 적합하다

6.1.2. MDO 응용 프로그래머 (MDO Application programmer)

DisMDO의 응용 프로그래머의 역할은 필요한 MDO 방법론을 DisMDO 프레임웍 상에 구현하는 역할을 한다. MDO 방법론은 템플릿 (template) 형태로 구현되어 최종사용자가 선택하여 사용할 수 있게 된다. 최종사용자는 이러한 템플릿을 사용자별로 용도에 맞게 인수 값들을 설정하여 사용하게 된다. 또한 DisMDO가 제공하는 인터페이스와 유사한 CAE 또는 Optimizer 인터페이스는 MDO 응용 프로그래머에 의해서 추가될 수 있다, 따라서 MDO 응용 프로그래머는 설계 도구들에 대한 전문지식과 프레임웍에 대한 개괄적 지식을 바탕으로 최종사용자의 수요에 대응하여 프레임웍의 적용 범위를 확장하는 역할을 하게 된다.

6.1.3. MDO 관리자 (MDO Administrator)

MDO 관리자의 주요 업무는 MDO 사용자들에게 적절한 권한의 사용자 계정을 부여하고 관리하는 기능과 DisMDO 시스템에서 발생하는 다양한 데이터를 백업하고 보호하는 일이다.

6.1.4. MDO 프레임웍 개발자 (MDO Framework Developer)

MDO 프레임웍 개발자는 MDO 응용 프로그래머가 DisMDO에 쉽게 구현하기 어려운 새로운 MDO 방법론을 구현하기 위해서 프레임웍 차원에서 지원하는 역할을 수행한다. 프레임웍 개발자는 프레임웍 구조 및 인터페이스에 대한 총괄적 지식과 특히 스크립트를 이용한 프레임웍 프로그래밍 기술이 필요하다.

6.2. 목표 시스템의 특징

DisMDO 프레임웍을 개발하는데 있어서 프레임웍이 갖춰야 할 기본적인 성격을 토대로 개발하는 것이 바람직하다. 이에 필요한 성격을 고려해 본다.

다양성: MDO는 정의상 다양한 해석 및 최적화 도구들을 사용할 필요가 있으며 이들은 또한 지리적으로 분산되어 있을 수 있다. 이러한 특징은 분산컴퓨팅 환경을 필요로 하며 다양한 컴퓨팅 플랫폼 (platform) 지원을 요구한다. 따라서 분산환경을 지원하는 CORBA나 Java RMI 등을 이용할 필요가 있으며 나아가 각 도구를 대표하는 에이전트를 개발하는 방안을 고려할 수 있다.

호환성: 독립적으로 개발된 CAD 및 CAE 등은 독자적인 데이터 양식을 가지고 있다. 이들 데이터 간의 호환성을 제공하기 위해서는 STEP이나 XML과 같은 국제적 산업체 표준 양식과 교환 방식을 채택할 필요가 있다.

효율성: 소프트웨어 공학적 시스템 개발 방법론의 적용을 통하여 개발 효율 뿐만 아니라 사용 효율도 추구할 수 있다. 특별히 객체지향 시스템 개발 방법론의 표준으로 부각된 Unified Modeling Language(UML)을 구현한 CASE 도구를 적극활용하여 효율성을 추구할 수 있다.

확장성: 새로운 CAD나 CAE 도구들을 수용하고 새로운 데이터 양식을 손쉽게 접목시킬 수 있는 기능을 말하는 것으로서 객체지향방법론의 속성계승 및 속성 오버라이딩 기능을 활용하고 각 도구와 데이터 표준을 대표하는 에이전트와 이들 간의 상호작용에 의한 시스템 구성으로 확장성을 확보할 수 있다.

확대성: 새로운 도구나 데이터를 수용하는 확장성과 달리 대규모 및 소규모 MDO 문제를 모두 만

족할 수 있는 성질을 말한다. 단일 컴퓨터에서 이루어지는 소규모 문제와 분산된 환경에서 이루어지는 대규모 문제에 모두 적합한 공통 인터페이스를 사용함으로써 확대성 문제에 대응할 수 있다

편의성: 지능형 사용자 인터페이스를 통한 편리한 시스템 사용 역시 필수적 항목이다. 특히 MDO 응용 분야에 따라서는 그래픽 사용자 인터페이스(GUI)를 사용한 반복적인 작업이 비효율적일 수 있음을 고려하여 GUI 작업의 결과는 사용자가 쉽게 알아보고 수정 또는 프로그램할 수 있는 문자 스크립트(script)로 대응되도록 하여 이들 간의 기능적 동치를 제공하는 것이 편의성을 증진에 도움이 될 수 있다.

6.3. DisMDO의 구성

DisMDO 프레임웍 설계의 기본 원칙은 사용자로 하여금 선언적 문제 정의(declarative specification)를 할 수 있도록 하는 것이다. 이와 반대되는 개념의 문제 정의 방법은 풀이절차 중심 문제 정의 (procedural specification)라 할 수 있다. 선언적 문제 정의에서는 해결하고자 하는 문제 자체, 즉 what에 해당되는 부분을 중심으로 문제를 정의하고 풀이절차 중심 문제 정의에서는 문제를 해결하는 방법과 순서, 즉 how에 해당하는 부분에 주안점을 두고 있다. DisMDO의 최종사용자는 선언적 문제 정의에 초점을 맞추고 풀이절차는 가능한 한 MDO 응용 프로그래머에 의해 제공된다.

DisMDO 프레임웍은 또한 재사용성을 고려한 컴포넌트(component)들로 구성된다. 이들 컴포넌트들은 프레임웍 구성 규칙에 의하여 결합되어 제한된 수의 컴포넌트들로 다양한 구성을 이룰 수 있게 된다. 또한 DisMDO는 일반 분산객체시스템 구성에 응용되는 3-tier 구조를 바탕으로 구성되어 Model (Data Store), View (GUI), Controller (Logic)의 분리를 이루고 이로부터 모듈간의 독립성, 모듈간의 호환성 및 부하 균등 배분의 장점을 활용할 수 있다.

DisMDO 프레임웍은 일반 소프트웨어 시스템의 개발 요소인 요구 사항 분석, 설계, 시스템 구성 요소(components) 식별, 시스템 구성 요소 간의 관계(relationship) 식별, 시스템 언어 및 GUI 설계 등의 체계적 적용하여 목표 시스템 특징을 추구하고 있다.

6.4. DisMDO 프레임웍 구조

DisMDO 프레임웍은 크게 MDO 커널(kernel), 데이터 서버, Optimizer 인터페이스, CAE 인터페이스의 네 부분으로 구성된다.

6.4.1. MDO 커널 (kernel)

MDO 커널은 MDO 문제 해결 과정으로 제시된 문제 정의, 프로세스 구성, 실행 및 감시, 후처리 등을 총괄하는 부분이다. 사용자는 커널 사용자 인터페이스를 이용하여 문제를 정의하고 프로세스를 구성하여 커널과 상호 작용을 하게 된다.

6.4.2. 데이터 서버

데이터 서버는 데이터베이스와 연계되어 정의된 문제와 프로세스를 저장하고 실행시 발생하는 데이터들을 체계적으로 관리한다.

6.4.3. Optimizer 인터페이스

DisMDO 프레임웍의 optimizer 인터페이스는 먼저 모든 optimizer에 공통적인 성격을 정의하고 이를 기반으로 한 optimizer 간의 계층을 형성하여 새로운 optimizer도 특이점만 표현하여 프레임웍에 비교적 손쉽게 연계하여 사용할 수 있게 된다.

6.4.4. CAE 인터페이스

Optimizer 인터페이스와 마찬가지로 모든 CAE는 공통성을 기반으로 한 CAE 간의 계층을 형성한다.

커널, Optimizer, CAE 인터페이스와 같이 사용자와 상호작용이 필요한 구성 요소들은 모두 사용자 인터페이스(UI)를 가지고 있다. 이들은 독립적인 프로세서로 작동하여 커널이나 optimizer 또는 CAE가 작동하는 컴퓨터와는 별도의 컴퓨터에서 사용자 입력을 받고 또한 결과를 출력할 수 있는 구조를 가지고 있다. 이들은 또한 시스템이 제공하는 기본 값을 변경하고자 할 때만 사용자 입력을 요구하여 사용자 편의성을 도모한다.

특별히 optimizer와 CAE를 위한 인터페이스는

다양한 상업용 또는 이미 개발되어 사용중인 소프트웨어와 효과적을 연계 기능을 제공하여야 한다. 또한 새로운 도구들을 손쉽게 수용할 수 있어야 한다. 이를 위하여 DisMDO는 인터페이스 제조기(interface factory)를 제공한다. 인터페이스 제조기는 인터페이스를 생성하여 역할을 한다. 예를 들어 파일 중심의 데이터 입출력을 하는 CAE를 위해서는 CAE의 입력 파일과 출력 파일을 분석하고 조작하는 일반적인 방법들을 제공하여 새로운 파일 중심 데이터 입출력 CAE 인터페이스를 사용자가 생성할 수 있다. 스크립트나 API를 이용하는 CAE나 optimizer도 유사하게 인터페이스 제조기를 구성할 수 있게 된다.

7. 요약 및 결론

MDO 방법론이란 MDO 프레임웍을 통하여 MDO 프로그램으로 구현될 설계자의 구상을 일컫는다. 이러한 MDO 방법론은 반완성 MDO 프로그램 이라고 볼 수 있는 템플릿(template)으로 존재할 수 있다. MDO 프로그램으로 구현될 이러한 MDO 방법론은 MDO 프레임웍에서 제공하는 MDO 프로그래밍 언어로 표현된다. 이러한 언어는 일반 프로그램밍 언어와 같은 문자 중심 언어로 표현할 수 도 있으며 그래픽 중심 언어로 표현할 수도 있다. MDO 프로그램은 CAE, optimizer등의 도구들과 이들을 이용한 실행 순서 및 선택조건, 반복조건, 병렬수행 등을 지정하는 제어문을 이용하여 작성된다.

MDO 프레임웍은 설계 및 해석 도구들간의 인터페이스를 제공하고, 이들 도구들이 사용하는 설계 데이터를 효율적으로 공유할 수 있도록 지원하여, 설계 작업을 정의, 실행, 관리하는 역할을 한다. 이러한 MDO 프레임웍은 설계 작업을 통합적으로 관리하고 자동화하여 설계 도구간의 데이터 전달과 변환에 소요되는 부담을 없애고, 다분야 전문가가 참여하는 공통 작업 환경을 제공함으로써 설계 효율성을 증진시킨다. 분산형 지능 시스템으로 구성된 DisMDO 프레임웍은 차세대 설계 기술로서 주목 받는 MDO 기술을 실현하여 설계 기술의 발전과 이에 따른 산업 발달에 직접적으로 기여할 것으로 기대된다.

8. 후기

이 연구는 한국과학재단 지정 최적설계신기술연구센터의 지원에 의해 수행되었습니다.

9. 참고문헌

[Korte 98] J. J. Kporte, R. P. Weston, and T. A. Zang, "Multidisciplinary Optimization Methods for Preliminary Design",

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

[Sobi 96] J. Sobieszczanski-Sobieski, R. T. Haftka, "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments", AIAA, 1996.

[Krish 98] R. Krishman, "Evaluation of Frameworks for HSCT Design Optimization", NASA/CR-1998-208731, 1998.

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

[iSight 99] Engineous Software, Inc., "iSight Technical Overview", 1999.

[IMAGE 97] M. A. Hale, "IMAGE User's Manual Version 1.5", 1997.