An evaluation of three commercially available integrated design framework packages for use in the Space Systems Design Lab

by Andrew Thomas Scott
submitted to Dr. John Olds
April 26th, 2001

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

APAS	Aerodynamic Preliminary Analysis System
CABAM	Cost and Business Analysis Module
DOE	Design of Experiments
DOT	Design Optimization Tools
DSM	Design Structure Matrix
GE	General Electric
GUI	Graphical User Interface
IDEAS	Integrated Design Engineering Analysis Software
IPREP	Interplanetary Preprocessor
ISE	Intelligent Synthesis Environment
MDO	Multi-disciplinary Design Optimization
MDOL	Multi-Disciplinary Optimization Language
POST	Program to Optimize Simulated Trajectories
RaDEO	Rapid Design Exploration and Optimization
RLV	Reusable Launch Vehicle
SCORES	SpaceCraft Object-oriented Rocket Engine Simulation
SCREAM	Simulated Combined-Cycle Rocket Engine Analysis Module
SHABP	Supersonic/Hypersonic Arbitrary Body Program
SSDL	Space Systems Design Lab
TCAT	Thermal Calculation Analysis Tool
UDP	Unified Distributed Panel

INTRODUCTION

In the design of any complex system, it customary to find that the decomposition of such a system divides the analysis of the problem into several smaller sections that focus on applicable engineering disciplines. This approach to design has not only been well established but, for very large systems, is the only realistically feasible direction to take. Unfortunately, it is extremely rare to find a problem that can be divided in such a way that the inputs and outputs of the disciplinary analyses do not heavily rely on one another. Because of the natural coupling that will arise between different disciplines, the design problem requires not only the system level requirements are met, but also that the results of each discipline are consistent with one another. Figure 1 illustrates the typical design structure matrix (DSM) used by the Space Systems Design Lab (SSDL) for its space vehicle design work.

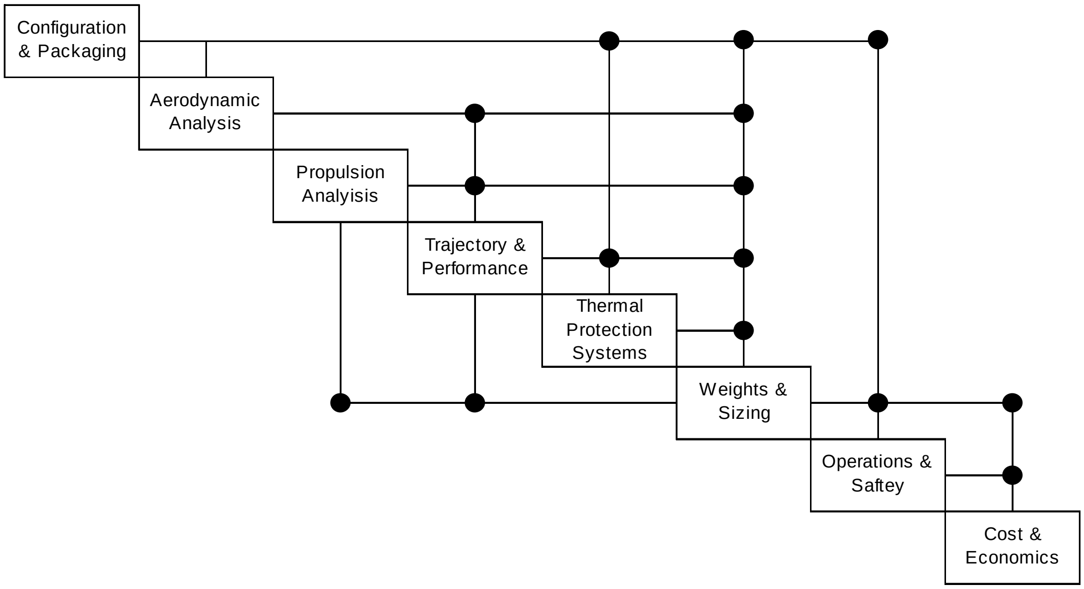


Figure 1: Design Structure Matrix for a typical space launch vehicle

Within the past decade, significant improvements have been made in the various methods by which these problems are typically approached. The simplest and earliest method is by simple iteration amongst the disciplines. Using reasonable guesses as a good starting point, analysis begins with one discipline, which, in turn, passes its verified results to the other disciplines that require them. Whenever the results of one discipline affect the inputs of a previously run discipline, iteration is required until the coupling variables converge to within acceptable tolerances. In traditional engineering environments, this iteration is accomplished by each disciplinary expert physically receiving his appropriate inputs, performing his analysis, and then physically giving the results to the other experts that require them. Depending on the number and fidelity of the individual disciplines, this process can take a great deal of time to acquire a single converged design. When optimization schemes are employed at the system level, the number of design points that must be evaluated grows quite large, and the overall design process can easily take months of time.

Figure 2: Early Design Practices

Following the well know paradigm that "time is money," it is a natural goal to want to reduce the turn-around time of the design process as much as possible. Towards this effort, several innovations have been made. The speed with which disciplinary analyses can be performed is steadily increasing as improvements in processing speed are constantly being made. Advanced optimization schemes reducing or eliminating the amount of disciplinary iteration have been developed [1]. Additionally, the advent of the Internet and advancements in electronic communications have made the transfer of information between engineers extremely fast and accurate. However, with all these advances a new bottleneck in the design process has arisen . . . organization. With all these improvements the amount of information that must be maintained, recorded and distributed has quickly become quite cumbersome. It is with these issues in mind that efforts have been made toward the development of integrated design frameworks that handle the collection, organization & distribution of information amongst various disciplinary analyses. Common to all such computational frameworks is an interface through which the user can define the flow of information between disciplines as well as how and when disciplinary analyses are run. Once these interrelationships are defined, it is logistically much easier for the engineer to perform a wide variety of system level design analyses, such as optimizations, trades and response surface approximations.

There have been many attempts to develop good computational frameworks. Some earlier implementations have been specific to a single type of problem with interfaces to various analyses more or less “hard-wired”. While this kind of framework might be very good for the kind of design problem it was created for, this type of implementation is typically inflexible in what it can do. As the development of such a framework is a technically difficult and time consuming process, such problem-specific frameworks are usually not the most realistic solutions as they themselves take a great deal of time and knowledge to develop. Recently, various companies in the computer software industry have developed integrated design framework software that is not restricted to any particular kind of design problem. By providing an easy means for the user to integrate their own disciplinary analyses, these computational frameworks allow the engineer to organize the design process of nearly any type of problem.

The SSDL within the Georgia Institute of Technology, does a wide variety of conceptual space-vehicle design using several in-house-developed disciplinary analysis tools and industry standard legacy codes. In order to improve the efficiency and level of design done in the lab, use of three commercially available integrated design framework packages is investigated and evaluated. These packages are Phoenix Integration’s ModelCenter, Engenious Software’s iSIGHT, and Technosoft’s AML. In this evaluation, consideration is given to a wide variety of criteria addressing the needs of the SSDL and key functionality of the software.

EVALUATION CRITERIA

Hardware Support

The bulk of the evaluation criteria for the candidate packages was based the specific and immediate needs of the SSDL. Within the SSDL there is a fairly heterogeneous collection of computer platforms and operating systems. Table 1 summarizes the various computers and platforms used in the lab. Most of these computers are designated as the personal computers of the lab members. The two SGI's, two PC's and one Mac are set aside for web hosting and for handling computationally intensive tasks. The chosen framework software should be able to run on as many of these machine/platform types as possible, while being able to easily interface and communicate with analytical tools across these platform boundaries.

Table 1: SSDL Hardware Breakdown

10	Gateway PC's running Windows NT
4	Dell PC's running Windows 2000
5	Macintosh Computers running MacOS
2	SGI - Unix Workstations

Disciplinary Analysis Integration

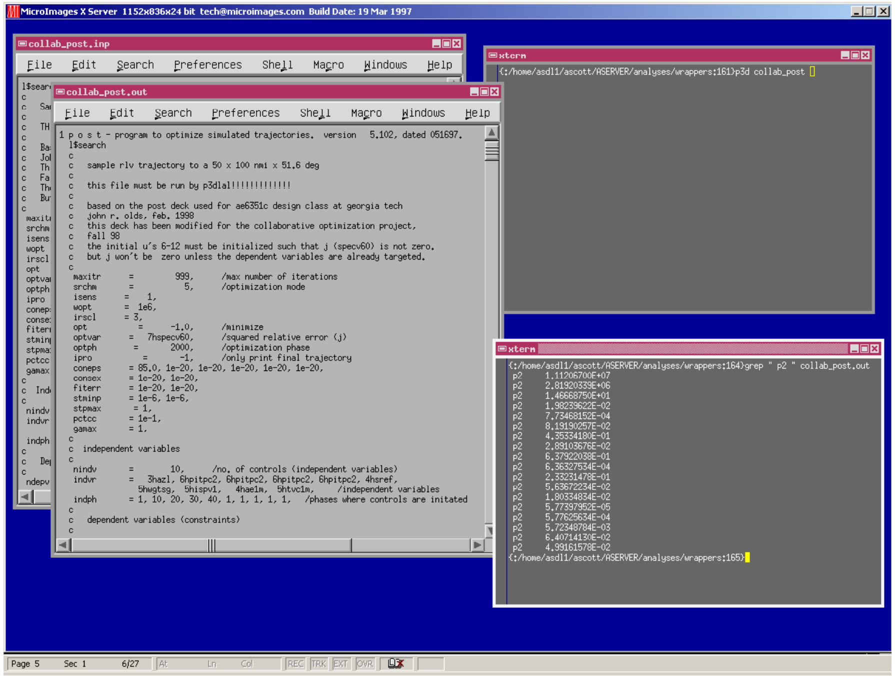
Most of the various disciplinary tools used in the SSDL can only be run on machines of a specific type. Table 2 summarizes information about the various disciplinary analysis tools used in the lab. It is a key requirement that the selected framework package be able to successfully interface with the various types of tools used in the SSDL. These tools can be categorized into one of three groups based on how the user

Table 2: Disciplinary Analysis Tools

Name	*Description*	*Platform*	*Type*
AATe	Operations	Windows/Mac	Excel based
APAS	Aerodynamics	UNIX	GUI Aplication
CABAM	Costs & Economics	Windows/Mac	Excel based
GT-Sizer	Weights & Sizing	Windows/Mac	Excel based
IDEAS	CAD Modeling	UNIX	GUI Application
IPREP	Interplanetary trajectories	any*	command-line
Miniver	TPS heat calculations	UNIX	command-line
POST	Trajectory & Performance	UNIX	command-line
SCORES	Liquid Rocket Engine Analysis	any*	command-line / Web
SCCREAM	Airbreathing Engine Analysis	any*	command-line / Web
TCAT	TPS stack sizing	any*	command-line

** indicates source code is available and that tool can be ported to any platform* interacts with that tool: command-line tools, Excel-based tools & GUI-tools. The selected framework package should be capable of integrating or "wrapping" most if not all types of these tools. Also, while the disciplinary tools will handle the bulk of analytical calculations, the user should be allowed define custom variables and perform simple calculations within the framework environment.

The first and most common group consists of command-line tools. These tools usually require the user to create or modify a specific input file, execute the analysis by typing in a command at a system prompt, and then parse a specific output file for the desired results. Some command-line tools will instead prompt the user directly for inputs and then output results to the screen, but these tools can nearly always be modified to operate using files instead. Though most command-line tools are fairly straightforward, some, like POST, require a good deal of user interaction in order to effectively use them. Because of this it is desired that the computational framework either allow the engineer to specify detailed rules and instructions on the use of that tool, and/or prompt the disciplinary expert responsible for that tool to validate the results. Most of the SSDL's in-house-developed command-line tools have also had web interfaces developed for them. It would be advantageous if the design framework had the ability to take advantage of these already-developed web interfaces.


MicroImages X Server 1152x836x24 bit tech@microimages.com Build Date: 19 Mar 1997
collab_post.inp
File Edit Search Preferences Shell Macro Windows Help
collab_post.out
File Edit Search Preferences Shell Macro Windows Help
xterm
xterm
Page 5 Sec 1 6/27 At Ln Col REC TRK EXT OVR

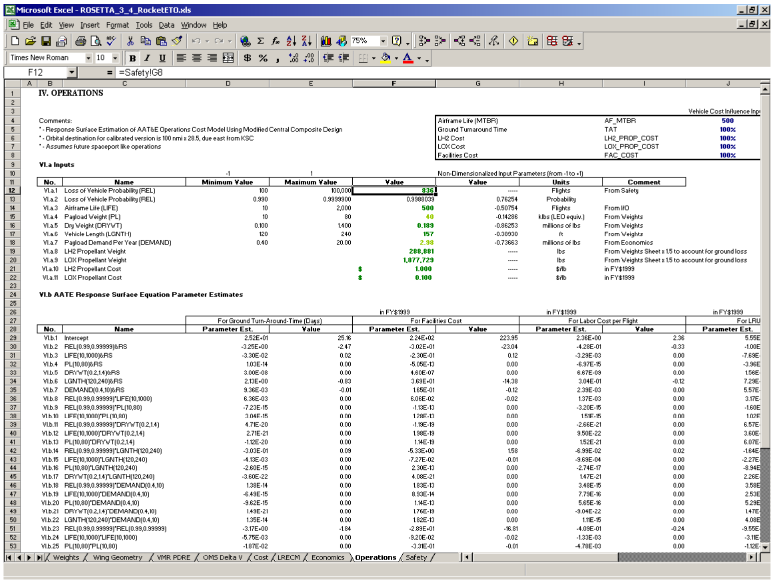
```
1 p o s t - program to optimize simulated trajectories.  version   5.102, dated 051697.
  l$search
   c
   c   sample rlv trajectory to a 50 x 100 nmi x 51.6 deg
   c
   c   this file must be run by p3dlal!!!!!!!!!!!!!
   c
   c   based on the post deck used for ae6351c design class at georgia tech
   c   john r. olds, feb. 1998
   c   this deck has been modified for the collaborative optimization project,
   c   fall 98
   c   the initial u's 6-12 must be initialized such that j (specv60) is not zero.
   c   but j won't be  zero unless the dependent variables are already targeted.
   c
    maxitr      =         999,      /max number of iterations
    srchm       =           5,      /optimization mode
    isens      =    1,
    wopt       =  1e6,
    irscl      = 3,
    opt           =      -1.0,      /minimize
    optvar      =   7hspecv60,      /squared relative error (j)
    optph        =       2000,      /optimization phase
    ipro           =       -1,      /only print final trajectory
    coneps      = 85.0, 1e-20, 1e-20, 1e-20, 1e-20, 1e-20,
    consex      = 1e-20, 1e-20,
    fiterr      = 1e-20, 1e-20,
    stminp      = 1e-6, 1e-6,
    stpmax       = 1,
    pctcc       = 1e-1,
    gamax       = 1,
   c
   c  independent variables
   c
    nindv       =        10,      /no. of controls (independent variables)
    indvr       =   3hazl, 6hpitpc2, 6hpitpc2, 6hpitpc2, 6hpitpc2, 4hsref,
                  5hwgtsg,  5hispv1,   4hae1m,  5htvc1m,     /independent variables
    indph       = 1, 10, 20, 30, 40, 1, 1, 1, 1, 1,   /phases where controls are initated
   c
   c   dependent variables (constraints)
   c
```

```
{:/home/asdl1/ascott/ASERVER/analyses/wrappers:161}p3d collab_post
```

```
{:/home/asdl1/ascott/ASERVER/analyses/wrappers:164}grep " p2 " collab_post.out
 p2     1.11206700E+07
 p2     2.81920339E+06
 p2     1.46668750E+01
 p2     1.98239622E-02
 p2     7.73468152E-04
 p2     8.19190257E-02
 p2     4.35334180E-01
 p2     2.89103676E-02
 p2     6.37922038E-01
 p2     6.36327534E-04
 p2     2.33231478E-01
 p2     5.63672234E-02
 p2     1.80334834E-02
 p2     5.77397952E-05
 p2     5.77625634E-04
 p2     5.72348784E-03
 p2     6.40714130E-02
 p2     4.99161578E-02
{:/home/asdl1/ascott/ASERVER/analyses/wrappers:165}
```

Figure 3: Sample command-line tool

The remaining tools developed within the SSDL are spreadsheet based, using Microsoft Excel. Excel is particularly advantageous when developing analytical methods that will inherently require a great deal of iteration between simple mathematical formulas. Such is usually the case when performing weights and sizing calculations. Additionally, the built-in capabilities of Solver allow for easy disciplinary optimizations. When developing economic analysis tools, the use of Excel is an obviously natural choice. Therefore, in addition to enabling the easy integration of command-line based analyses, the computational framework should have the capability to easily incorporate Excel-based tools. This support should allow the user to specify inputs, macros to be called, and outputs to be retrieved.



F12 = =Safety!G8

IV. OPERATIONS

Comments:
* - Response Surface Estimation of AAT&E Operations Cost Model Using Modified Central Composite Design
* - Orbital destination for calibrated version is 100 nmi x 28.5, due east from KSC
* - Assumes future spaceport like operations

Vehicle Cost Influence Inp

Airframe Life (MTBR)	AF_MTBR	500
Ground Turnaround Time	TAT	100%
LH2 Cost	LH2_PROP_COST	100%
LOX Cost	LOX_PROP_COST	100%
Facilities Cost	FAC_COST	100%

VI.a Inputs

		-1	1		Non-Dimensionalized Input Parameters (from -1 to +1)		
No.	Name	Minimum Value	Maximum Value	Value	Value	Units	Comment
VI.a.1	Loss of Vehicle Probability (REL)	100	100,000	836	-----	Flights	From Safety
VI.a.2	Loss of Vehicle Probability (REL)	0.990	0.9999900	0.9988039	0.76254	Probability	
VI.a.3	Airframe Life (LIFE)	10	2,000	500	-0.50754	Flights	From I/O
VI.a.4	Payload Weight (PL)	10	80	40	-0.14286	klbs (LEO equiv.)	From Weights
VI.a.5	Dry Weight (DRYWT)	0.100	1.400	0.189	-0.86253	millions of lbs	From Weights
VI.a.6	Vehicle Length (LGNTH)	120	240	157	-0.38930	ft	From Weights
VI.a.7	Payload Demand Per Year (DEMAND)	0.40	20.00	2.98	-0.73663	millions of lbs	From Economics
VI.a.8	LH2 Propellant Weight			288,881	-----	lbs	From Weights Sheet x 1.5 to account for ground loss
VI.a.9	LOX Propellant Weight			1,877,729	-----	lbs	From Weights Sheet x 1.5 to account for ground loss
VI.a.10	LH2 Propellant Cost			$ 1.000	-----	$/lb	in FY$1999
VI.a.11	LOX Propellant Cost			$ 0.100	-----	$/lb	in FY$1999

VI.b AATE Response Surface Equation Parameter Estimates

				in FY$1999		in FY$1999		in FY$1999
		For Ground Turn-Around-Time (Days)		For Facilities Cost		For Labor Cost per Flight		For LRU
No.	Name	Parameter Est.	Value	Parameter Est.	Value	Parameter Est.	Value	Parameter Est.
VI.b.1	Intercept	2.52E+01	25.16	2.24E+02	223.95	2.36E+00	2.36	5.55E
VI.b.2	REL(0.99,0.99999)&RS	-3.25E+00	-2.47	-3.02E+01	-23.04	-4.28E-01	-0.33	-1.00E
VI.b.3	LIFE(10,1000)&RS	-3.30E-02	0.02	-2.30E-01	0.12	-3.29E-03	0.00	-7.69E-
VI.b.4	PL(10,80)&RS	1.03E-14	0.00	-5.05E-13	0.00	-6.97E-15	0.00	-3.96E
VI.b.5	DRYWT(0.2,1.4)&RS	3.00E-08	0.00	4.60E-07	0.00	6.67E-09	0.00	1.56E-
VI.b.6	LGNTH(120,240)&RS	2.13E+00	-0.83	3.69E+01	-14.38	3.04E-01	-0.12	7.29E-
VI.b.7	DEMAND(0.4,10)&RS	9.36E-03	-0.01	1.65E-01	-0.12	2.39E-03	0.00	5.57E-
VI.b.8	REL(0.99,0.99999)*LIFE(10,1000)	6.36E-03	0.00	6.06E-02	-0.02	1.37E-03	0.00	3.17E-
VI.b.9	REL(0.99,0.99999)*PL(10,80)	-7.23E-15	0.00	-1.13E-13	0.00	-3.20E-15	0.00	-1.60E
VI.b.10	LIFE(10,1000)*PL(10,80)	3.04E-15	0.00	1.28E-13	0.00	1.51E-15	0.00	1.02E
VI.b.11	REL(0.99,0.99999)*DRYWT(0.2,1.4)	4.71E-20	0.00	-1.19E-19	0.00	-2.66E-21	0.00	6.57E-
VI.b.12	LIFE(10,1000)*DRYWT(0.2,1.4)	2.71E-21	0.00	1.98E-19	0.00	9.50E-22	0.00	3.60E-
VI.b.13	PL(10,80)*DRYWT(0.2,1.4)	-1.12E-20	0.00	1.14E-19	0.00	1.52E-21	0.00	6.07E-
VI.b.14	REL(0.99,0.99999)*LGNTH(120,240)	-3.03E-01	0.09	-5.33E+00	1.58	-6.99E-02	0.02	-1.64E
VI.b.15	LIFE(10,1000)*LGNTH(120,240)	-4.13E-03	0.00	-7.27E-02	-0.01	-9.69E-04	0.00	-2.27E-
VI.b.16	PL(10,80)*LGNTH(120,240)	-2.60E-15	0.00	2.30E-13	0.00	-2.74E-17	0.00	-8.94E
VI.b.17	DRYWT(0.2,1.4)*LGNTH(120,240)	-3.60E-22	0.00	4.08E-21	0.00	1.47E-21	0.00	2.26E-
VI.b.18	REL(0.99,0.99999)*DEMAND(0.4,10)	1.38E-14	0.00	1.83E-13	0.00	3.48E-15	0.00	3.58E
VI.b.19	LIFE(10,1000)*DEMAND(0.4,10)	-6.49E-15	0.00	8.93E-14	0.00	7.79E-16	0.00	2.53E
VI.b.20	PL(10,80)*DEMAND(0.4,10)	-9.62E-15	0.00	1.14E-13	0.00	5.65E-16	0.00	5.29E
VI.b.21	DRYWT(0.2,1.4)*DEMAND(0.4,10)	1.49E-21	0.00	1.76E-19	0.00	-9.04E-22	0.00	1.47E-
VI.b.22	LGNTH(120,240)*DEMAND(0.4,10)	1.35E-14	0.00	1.82E-13	0.00	1.11E-15	0.00	4.08E
VI.b.23	REL(0.99,0.99999)*REL(0.99,0.99999)	-3.17E+00	-1.84	-2.89E+01	-16.81	-4.09E-01	-0.24	-9.55E-
VI.b.24	LIFE(10,1000)*LIFE(10,1000)	-5.75E-03	0.00	-9.20E-02	-0.02	-1.33E-03	0.00	-3.11E
VI.b.25	PL(10,80)*PL(10,80)	-1.87E-02	0.00	-3.31E-01	-0.01	-4.78E-03	0.00	-1.12E-

Figure 4: Sample Excel-based tool

The final class of analytical tools used in the SSDL is comprised of the complex commercial and legacy tools that require a high level of user interaction through a GUI interface. These include CAD programs such as IDEAS and aerodynamic analyses tools like APAS. Because these analyses take the longest to perform, these analyses are usually taken out of the main design loop by running them once and then appropriately scaling their results. The effects of geometry-dependent design parameters that do not scale well, such as packaging efficiency, are usually accounted for by response surface approximations. This approach requires that these analyses be performed only a few times. Since these tools require such a high level of user interaction, automation of these disciplines within a computational framework would most likely be quite difficult. Furthermore, as the SSDL's traditional design methodology already isolates these types of tools from the main design loop (see Figure 1), the ability to integrate them within the computational framework is not of the highest priority, nor necessarily even required. This kind of capability would be greatly advantageous and necessary in higher-fidelity design work where the design process requires constantly updated geometric and aerodynamic information. As most of the SSDL's work is at the conceptual and preliminary design level, this level of analysis is not usually needed.

A final requirement concerning tool wrapping should address the reusability of wrapped tools. It is desirable that once a code had been wrapped that the work required to reuse that code be kept to a minimum. This kind of flexibility is a key capability that distinguishes a good computational framework from inflexible, problem specific framework solutions.

Disciplinary Analysis Distribution

The next set of evaluation criteria is concerned with how the framework is capable of interacting with analysis codes once they are wrapped. The rational behind these requirements is derived from the SSDL's ideology on good design group organization. As mentioned before, complex design problems are nearly always divided into several smaller disciplinary problems. The advantage of this strategy is that it allows engineers to be responsible for the disciplinary sub-problems to which their expertise is most applicable. This allows the engineer to better focus his research and tool-development efforts in a specific field, by not required him to have as extensive a working knowledge of other disciplines. This strategy of distributed design not only ensures quality engineering in each discipline but also lends itself to a very natural distribution of work. It is desired that the framework software take advantage of this well-established organization of labor and expertise.

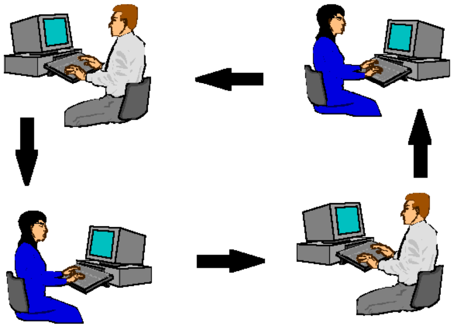


Figure 5: Distributed Design

Figure 6: Monolithic Codes

Efforts have proven that trying to automate and compile the analyses of every discipline into a single "monolithic" analysis tool has many disadvantages. Such an effort removes the distributed team of experts from the design process. This requires that the system-level user be either an unrealistic "super-engineer" or simply disadvantaged by not be able to readily identify and correct problems that a disciplinary expert could. Additionally the development of such monolithic codes, is quite cumbersome and usually produces very problem-specific solutions.

A good computational framework should combine the efficiency and speed of monolithic codes with the practical distribution of labor and expertise found in distributed design. A well-integrated design framework should be capable of allowing the disciplinary analyses to be run on the experts' machines under their supervision, while the overall design process is coordinated and directed by a central system-level user. In this way, the disciplinary experts could locally maintain their analytical tools, and, the computational load of the design would be distributed over several machines. To further keep the experts "in the loop" the best framework implementation would allow the disciplinary expert to monitor, verify, modify and approve the results of his tool before they are used returned to the system level operations of the framework. Naturally, the option to allow disciplinary tools to run unattended should also be available.

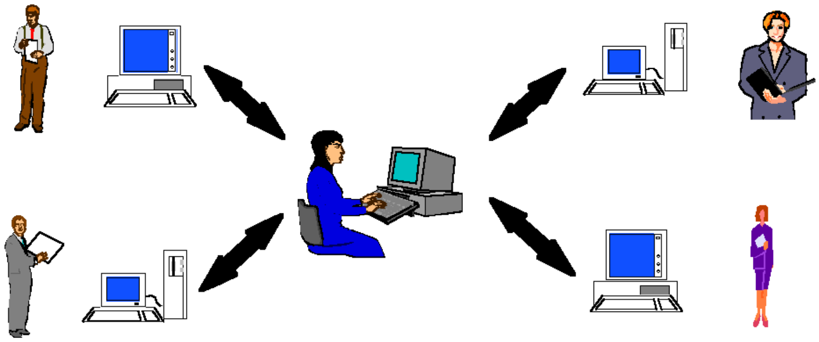


Figure 7: Integrated Design Framework

In addition to the accommodating the needs of the more-traditional methods used to organize design problems, the design framework should be able to take advantage of the computational benefits provided by advanced Multi-disciplinary Design Optimization (MDO) methods [1]. Specifically, the framework should be capable of executing distributed tools in parallel. Not only does this capability speed up the overall process, but also distributes the computational load amongst numerous machines. In addition to allowing different analyses to run in parallel, the framework should be able to thread several instances of the same tool to run at once. This capability should enable the user to simultaneously analyze multiple design points from a Design of Experiments (DOE).

System Level Analysis

With the means to wrap and distribute analytical tools at the disciplinary level fully considered, attention must be turned to the computational framework's capabilities at the system level. The desired framework should have a wide variety of analytical schemes available at the system level. The range of methods available should include support for zero and first-order optimization methods, DOE methodologies, and stochastic methods such as Genetic Algorithms and Simulated Annealing. Additionally, there should be an easy means by which the user can supply and integrate his own optimization algorithms. This ability should range from simple scripting for convergence loops to an easy means to integrate external optimization codes. As with any optimization problem, modifications are usually required to the internal parameters of the algorithm; the included optimization methods should have as many of these parameters available to the user as possible. The option for the user to supply gradient information directly to the optimizer should also be available. Response Surface methods should be capable of determining the required array of design point analyses, while also allowing the user to supply his own. The SSDL also uses a good deal of probabilistic simulations in its various studies; therefore, the inclusion of Monte Carlo simulation capabilities in the framework would be a great asset.

Data Collection

In addition to final results, there is great deal of information that the user should be able to track and record during any system level analysis. At the basic level the computational framework should keep track of any and all changes made to the system-level variables, disciplinary coupling variables, constraints and objectives. The user should be able to save and load convergence histories, DOE runs and simple design point evaluations for later retrieval, analysis and comparison. Additionally, the user should be able to custom configure what data is recorded and when. Because, things invariably will go wrong, the framework package should provide the ability to stop an optimization process and restart it either from the last or any previous point in the iteration history. Built-in graphing capabilities should be quite extensive providing the user with a wide range of options. Finally, all information should be stored in a format that is easily exported to external programs such as Excel.

Usage

The final group of evaluation criteria looks at the use of the candidate framework, primarily, how easy it is to use and learn. Considerations include overall user-friendliness, the ease with which tools can be wrapped and distributed, and the difficultly or lack thereof in setting up and performing system-level analysis. The product should be easy to learn, providing good documentation and several example problems and/or tutorials. The availability and quality of customer support is also an important consideration. As part of an institute of higher learning and training, knowledge and proficiency with the product should be a marketable and valuable skill for the lab members. This implies that the framework package should be well supported and used by several Governmental and Commercial organizations in the space industry. Finally, as the SSDL operates under a budget, due consideration must be given to what if any the cost will be for obtaining and using the product.

EVALUATION METHODOLGY

So that the evaluation & comparison is as fair and thorough as possible, attempts were made to implement a common design problem in each of the candidate frameworks. The problem chosen was to find the optimal configuration for a single-stage-to-orbit reusable launch vehicle (RLV). The analysis of this problem was simplified to involve only three disciplines: trajectory & performance, propulsion, and weights & sizing. A vehicle configuration & geometry was used for which aerodynamic data was already available. POST, which was used for the trajectory analysis, is an industry standard legacy code that is run by creating and modifying a detailed input file, commonly referred to as an input deck. Results from POST are written into a similarly detailed output file. In the SSDL, POST is set up to run in a command-line fashion on either of two SGI workstations. For the propulsion analysis, an in-house developed tool, SCORES, was chosen. This code has a web interface, but can also be run at the command line on either of the two Unix machines. Finally, the weights and sizing was performed by another in-house tool, GT-Sizer, which is Excel-based and uses the built-in Solver to photographically rescale the vehicle. This problem, while simple, exhibits most of the key functional requirements that the SSDL would expect of a good frameworks package. The disciplinary tools used, represent industry standard legacy codes and custom in-house-developed tools. This set of tools must also be run on different machines and platforms. Additionally, it is a design problem with which the lab is very familiar, making it well suited as a test case for whatever optimization and system-level analysis capabilities are provided in the framework package. The chart on the following page summarizes the conclusion of the evaluation. Detailed explanations of the evaluation are provided for each candidate in the sections that follow.

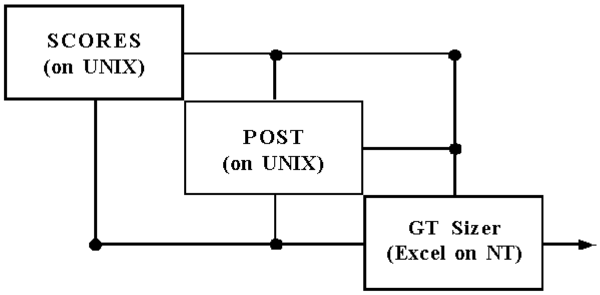


Figure 8: Design Structure Matrix of simplified RLV test problem

Integrated Design Framework Criteria List

🪐	Excellent	☾	Poor
●	Good	○	Non-Existant
◖	Fair	–	Could not be evaluated

	Model Center	ISight	AML
Operating System Support			
Unix - SGI	●	🪐	🪐
Windows NT/2000	🪐	🪐	🪐
Macintosh OS	◖	○	○
Disciplinary Level			
<u>Analysis Integration</u>			
Support for easily defined internal calculations	●	🪐	☾
Support for command-line tools	🪐	🪐	–
Support for web-based tools	○	○	–
Support for Excel based tools	🪐	◖	–
Support for GUI/CAD applications	☾	☾	◖
Reusability of wrapped tools	🪐	●	–
<u>Analysis Distribution</u>			
Capability to run analysis locally	●	🪐	–
Ability to distribute analyses to other machines	🪐	☾	–
Ability to automate the execution of analyses	●	🪐	–
Ability for end user to monitor/verify/modify & approve results	◖	☾	–
Ability to run several different analyses in parallel	○	🪐	–
Ability to simultaneously run several distributed instances of the same analysis	○	●	–
System Level			
<u>Analysis Methods</u>			
Easy set up of the design structure matrix and data-flow	●	🪐	–
Zero-order search methods	○	○	–
1st-order search methods	●	🪐	–
Design of Experiments / Response Surface Methods	●	🪐	–
Stochastic methods	○	🪐	–
Monte Carlo Simulations	○	🪐	–
Easy means to integrate user-supplied optimizer	☾	☾	–
Ability for user to directly supply gradients	○	◖	–
Ability to accommodate discrete variables	○	🪐	–
<u>Data Collection</u>			
Ability to track all changes to all variables, objectives and constraints	●	🪐	–
Ability to save and restore convergence histories, DOE and point evaluations	🪐	●	–
Ability to start and stop and restart optimizations for previous point in iteration history	◖	☾	–
Data stored in an easily managed format	🪐	🪐	–
Extensive graphing capabilities	◖	🪐	–
Usage			
Overall User-friendliness	🪐	◖	☾
Short Learning Curve	🪐	◖	☾
Extensive, easy to use documentation	🪐	●	●
Tutorials & Examples available	🪐	◖	🪐
Availability & Quality of customer support	●	🪐	●
Wide use in Governmental & Commercial space industry	🪐	🪐	◖
Cost	🪐	🪐	🪐

MODELCENTER

ModelCenter is developed by Phoenix Integration, which is located in the Virginia Tech Corporate Research Center. Phoenix Integration was started by two Virginia Tech doctoral students, Scott Woyak and Brett Malone, and their mentor, Arvid Myklebust, a professor of mechanical engineering. Woyak was a PhD candidate in Mechanical Engineering working in a CAD lab on a software integration project funded by IBM. Malone was a PhD candidate in Aerospace Engineering doing research on computer-aided aircraft design for NASA in the ACSYNT (AirCraft Synthesis) Institute. The two teamed up with Myklebust to found Phoenix in 1995. The complete software package required to set up this integrated design framework actually includes a second auxiliary program, called Analysis Server. The latest release of the core program ModelCenter, version 3.0, is currently available for Windows NT only. Analysis Server, now in version 2.0, is written in Java, and is available in an already-ported form for both Windows NT and Unix platforms. A generic version of Analysis Server is also available which, because it is written in Java, can be ported to any machine platform, including Macintosh. This kind of installation was successfully done on an Macintosh running OS X. ModelCenter and the Analysis Server work seamlessly together, to allow the user to integrate disciplinary analyses across various platforms.

Perhaps the simplest way to describe how these two programs work together is by comparison to a similar and more familiar architecture. Analysis Server operates much like a Web Server, only instead of hosting web pages, the content it hosts are disciplinary analyses. The user sets up his analysis by creating a ".fileWrapper" file that specifies how input files are created, what executions must be performed, and how to parse output files for results. Once wrapped, the user can then "publish" this analysis by moving the ".fileWrapper" file to an appropriately designated folder.

Analogous to a Web browser, the ModelCenter program can access and utilize these wrapped analyses by negotiating with any number of Analysis Server programs being run on different machines. The machines running Analysis Server can be of different platforms types, effectively allowing the user to integrate analysis from multiple platforms into a single model. In testing, it was quite easy to run ModelCenter from outside the Georgia Tech network and then access and integrate analyses that were hosted by machines within the SSDL. By expansion, it should be possible to incorporate Analysis Server hosted analyses from anywhere in the world into a single, central ModelCenter model. Though security issues are not a large concern for the SSDL, it is possible to place restrictions on who is able to access and use the analyses hosted on an Analysis Server.

Analysis Server is not only capable of hosting command-line type analyses, but Excel-based analyses as well. Similar to the means to wrap command-line codes, the user creates an ".excelWrapper" file that contains information detailing the input and output cells as well as what Excel macros must be called. Additionally, the latest version of Analysis Server has included the ability to define different methods for Excel-based tools that can be used to call other macros within the Excel workbook as desired. The user also has the option to either make the Excel application visible in the foreground or run unobtrusively in the background. Using the Analysis Server to wrap Excel-based tools allows the user to run that analysis on machines other than local ModelCenter machine. However, the option is available to wrap an Excel spreadsheet directly in ModelCenter and run the program locally. Though the Analysis Server program can be installed on Macintosh system, the support for Excel-based tools is limited to the Windows version.

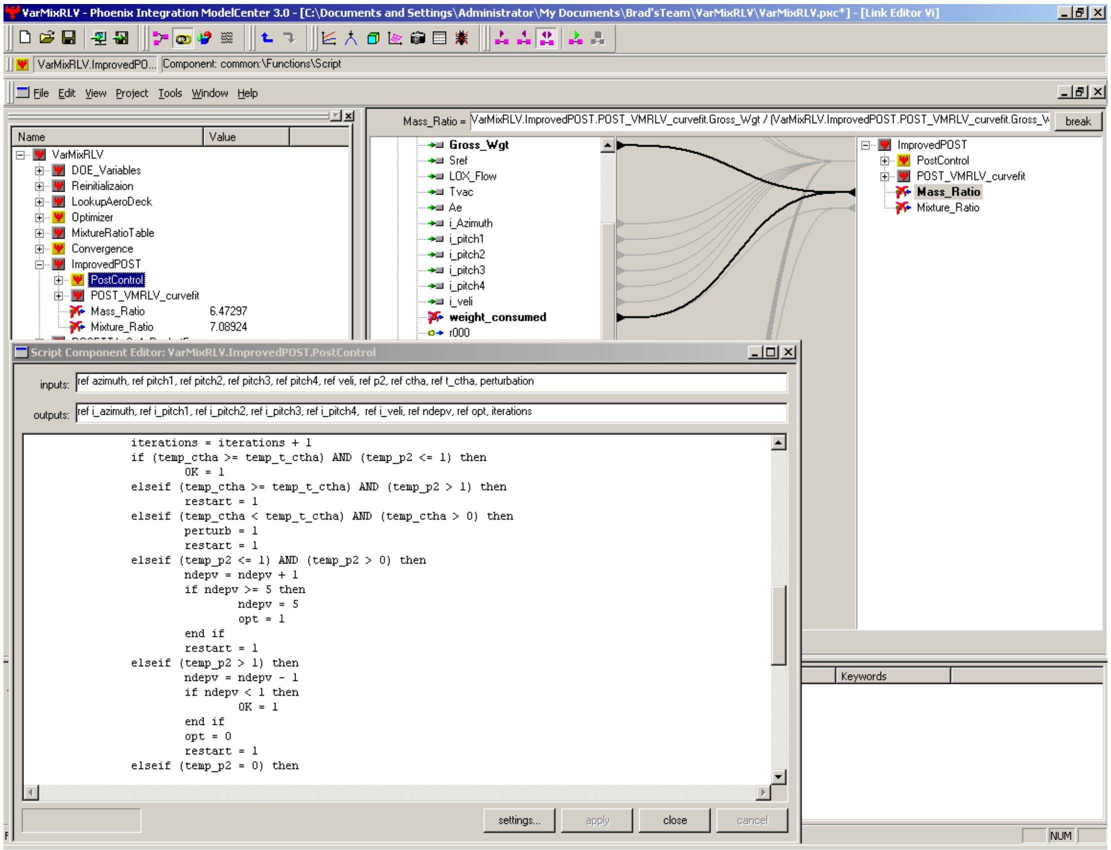
ModelCenter does advertise integration with Parametric Technology Corporation's Pro/Engineer CAD software. As the SSDL uses SDRC's I-DEAS CAD package for its geometric modeling, this feature was not examined. Additionally, there is some limited geometric modeling capabilities available in ModelCenter, but as this was not a major feature the SSDL was looking at, its potential uses were not fully explored. ModelCenter has also recently announced successful integration with Dassault Systemes' CATIA software, but again, this capability was not reviewed.

Through the use of the Analysis Server, ModelCenter makes reusing wrapped codes inherently a very easy task. Once a code is wrapped and set up on a specific machine, it simply a click and drag to add it to any new design project. If a particular disciplinary tool needs to be run on a different machine, it is simply a matter of moving the appropriate files and installing and starting the Analysis Server on that machine. The easy to Link Editor in ModelCenter allows the user to define the data flow between disciplines. Using the defined data flow, ModelCenter keeps track of variable dependency and can be configured to either automatically run an analysis whenever an input is changed, and/or when an invalid output is demanded. The most recent version of the Analysis Server has added the capability to prompt the user for verification of disciplinary results before continuing. This capability allows the disciplinary experts to review and if necessary modify the results of their analyses before they are sent to the system-level operations of ModelCenter. Unfortunately, this valuable capability seems to be limited to either operate in an “always pause” or “never pause” fashion. Ideally, the user should be able to also specify conditions under which the analysis will prompt the user for verification.

Many sophisticated optimization strategies, such as collaborative optimization, organize the data flow between the system level optimizer and the disciplinary analyses in such a way that the latter are allowed to run independently from one another in parallel. Unfortunately, the current version of ModelCenter does not possess the capability to either run different analysis in parallel or distribute several instances of single analysis to multiple machines.

The attempts to analyze the simplified RLV problem met with good success. Using ModelCenter, the Excel-based GT-Sizer code was wrapped and run on a Windows NT machine. POST and SCORES were wrapped and set up to run on each of the two Unix machines. All of the codes were easily integrated into a common model allowing for system-level analysis and optimization. Using this model, optimizations were performed on the design using a simple iterative method, a response surface approximation and collaborative optimization [2]. Both the simple iterative and response surface methods worked well and were far easier to set up and run with ModelCenter than without. Though successful application of collaborative optimization to this problem has not yet been achieved, the progress made so far would not have been realistically possible without some framework package such as ModelCenter.

At the system level, ModelCenter does possess some very attractive features, though the various means of analysis are somewhat limited. For example, ModelCenter has a limited number of built-in optimization schemes and lacks the ability to execute analyses in parallel. The Link Editor (see Figure 9) in ModelCenter provides a very easy to use "click and drag" approach to defining the flow of data between different analyses. Additionally, new variables can be easily introduced at the system level and nicely organized into appropriate assemblies. Input variables can either have their values directly specified or defined by formulas that use other variables as arguments. More complicated internal calculations can be accommodated using the script component. This script component can also be used to define more sophisticated methods of data flow such as convergence loops. Conveniently, the user has a wide variety of scripting languages to choose from when creating custom scripts.



Mass_Ratio 6.47297
Mixture_Ratio 7.08924

Mass_Ratio = VarMixRLV.ImprovedPOST.POST_VMRLV_curvefit.Gross_Wgt / (VarMixRLV.ImprovedPOST.POST_VMRLV_curvefit.Gross_W

inputs: ref azimuth, ref pitch1, ref pitch2, ref pitch3, ref pitch4, ref veli, ref p2, ref ctha, ref t_ctha, perturbation

outputs: ref i_azimuth, ref i_pitch1, ref i_pitch2, ref i_pitch3, ref i_pitch4, ref i_veli, ref ndepv, ref opt, iterations

```
        iterations = iterations + 1
        if (temp_ctha >= temp_t_ctha) AND (temp_p2 <= 1) then
                OK = 1
        elseif (temp_ctha >= temp_t_ctha) AND (temp_p2 > 1) then
                restart = 1
        elseif (temp_ctha < temp_t_ctha) AND (temp_ctha > 0) then
                perturb = 1
                restart = 1
        elseif (temp_p2 <= 1) AND (temp_p2 > 0) then
                ndepv = ndepv + 1
                if ndepv >= 5 then
                        ndepv = 5
                        opt = 1
                end if
                restart = 1
        elseif (temp_p2 > 1) then
                ndepv = ndepv - 1
                if ndepv < 1 then
                        OK = 1
                end if
                opt = 0
                restart = 1
        elseif (temp_p2 = 0) then
```

Figure 9: Custom Scripts and Variable Linking in ModelCenter

For system-level optimizations, ModelCenter has provided a front end to the popular and commercially available Design Optimization Tools (DOT) developed by Vanderplaats Research & Development, Inc. This is the very optimization package that the SSDL often uses in its optimizations. This package includes a variety of optimization methods as summarized in Table 3. The latest version of ModelCenter has allowed the user access to nearly all of the internal parameters available with these methods. Unfortunately, there is currently no built-in means to directly supply gradient information to the optimizer, leaving the analyst with finite difference methods as the only option for gradient calculations. Also, while ModelCenter does allow the user to specify internal variables as being integers, the built-in optimization tool is not designed to explicitly handle these kinds of variables.

Table 3: Optimization Algorithms available in ModelCenter

Name
Variable Metric Method (unconstrained)
Conjugate Gradient Method (unconstrained)
Method of Feasible Directions
Sequential Linear Programming
Sequential Quadratic Programming

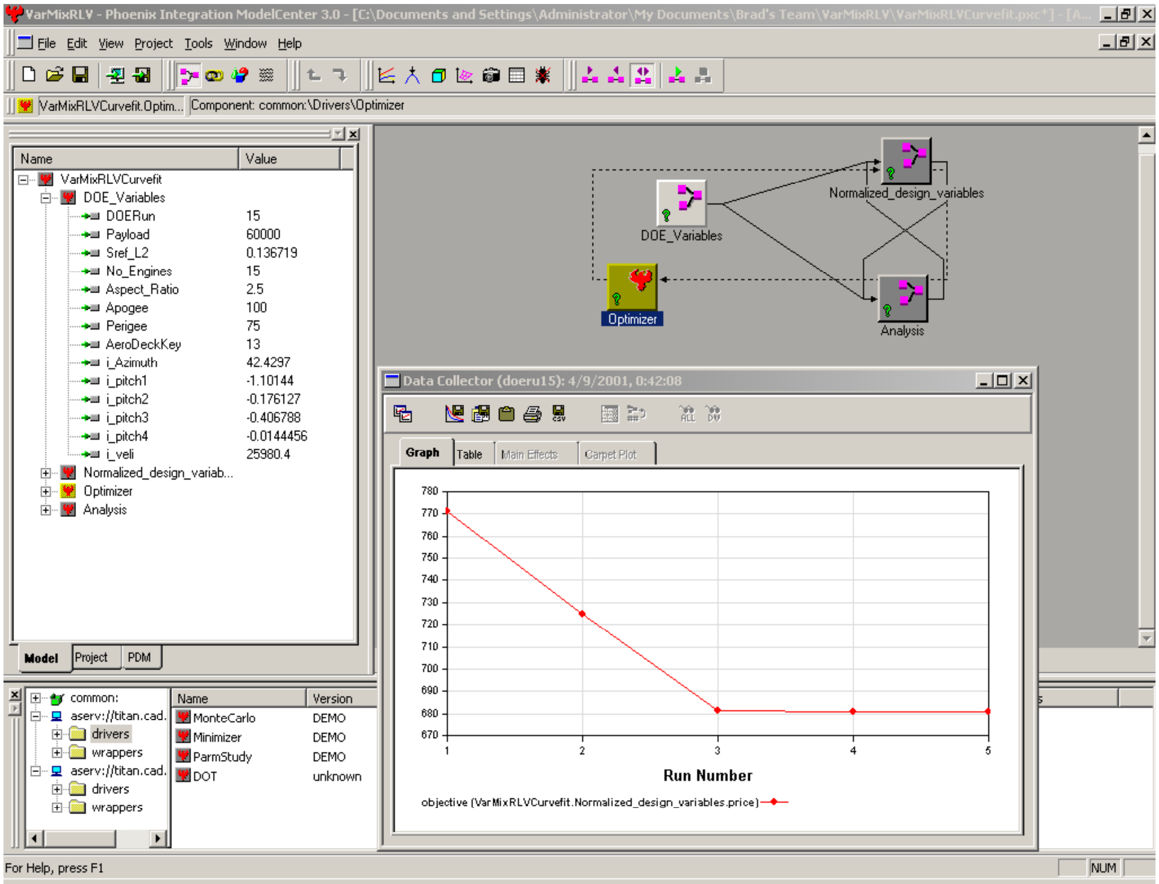
The capabilities provided for performing DOE studies in ModelCenter are quite good. A list of the options available is provided in Table 4. In addition to being able to automatically determine the array of design point evaluations required for the level of study desired, ModelCenter allows the user to supply his own series of DOE runs. A good number of different design arrays are available providing various levels of accuracy. Once analysis is complete, ModelCenter provides some limited options for analyzing the collected results.

Table 4: Design of Experiment options in ModelCenter

Name	*Level*
Full Factorial (any level)	any
1/2 Fractional Factorial Design	2
Foldover Design	2
Placket Burman Design	2
Taguchi L16 Design	2
Central Composite Deisgn	3
Face Centered Central Composite Design	3
Box Behnken Design	3

ModelCenter does not provide any Zero-order or Stochastic optimization methods, nor does it include any built-in capability to perform Monte Carlo simulations. However, the capabilities of the previously mentioned script component are extensive enough to allow the user to write complete custom optimization schemes. Additionally, one can integrate external optimization codes into ModelCenter using procedures specified in the documentation. This process, as it is somewhat more complicated, requires a good bit of programming know-how. However, it does work, and was successfully used to integrate the SSDL's version of DOT with the added functionality to directly supply gradients.

As any system-level analysis progresses, ModelCenter tabulates the values of all variables in the design problem. These values can easily be plotted vs. iteration number (see Figure 10) or each other. Any results can be saved and restored, and are easily exportable to Excel. At the click of a button the values of all variables can easily be restored to a previous state. Although an optimization process can be restarted from any point in the iteration history, it will unfortunately be a "cold-start" with any information that the optimizer might have collected about the design space up to that point being lost.


VarMixRLV - Phoenix Integration ModelCenter 3.0 - [C:\Documents and Settings\Administrator\My Documents\Brad's Team\VarMixRLV\VarMixRLVCurvefit.pxc*] - [A...
File Edit View Project Tools Window Help
DOE_Variables
Normalized_design_variables
Optimizer
Analysis
Graph
Table
Main Effects
Carpet Plot
Model
Project
PDM
780
770
760
750
740
730
720
710
700
690
680
670
1
2
3
4
5
Run Number
objective (VarMixRLVCurvefit.Normalized_design_variables.price)
NUM

VarMixRLVCurvefit.Optim... Component: common:\Drivers\Optimizer

Name	Value
VarMixRLVCurvefit	
DOE_Variables	
DOERun	15
Payload	60000
Sref_L2	0.136719
No_Engines	15
Aspect_Ratio	2.5
Apogee	100
Perigee	75
AeroDeckKey	13
i_Azimuth	42.4297
i_pitch1	-1.10144
i_pitch2	-0.176127
i_pitch3	-0.406788
i_pitch4	-0.0144456
i_veli	25980.4
Normalized_design_variab...	
Optimizer	
Analysis	

common:
aserv://titan.cad.
drivers
wrappers
aserv://titan.cad.
drivers
wrappers

Name	Version
MonteCarlo	DEMO
Minimizer	DEMO
ParmStudy	DEMO
DOT	unknown

Data Collector (doeru15): 4/9/2001, 0:42:08

For Help, press F1

Figure 10: Optimization run in ModelCenter

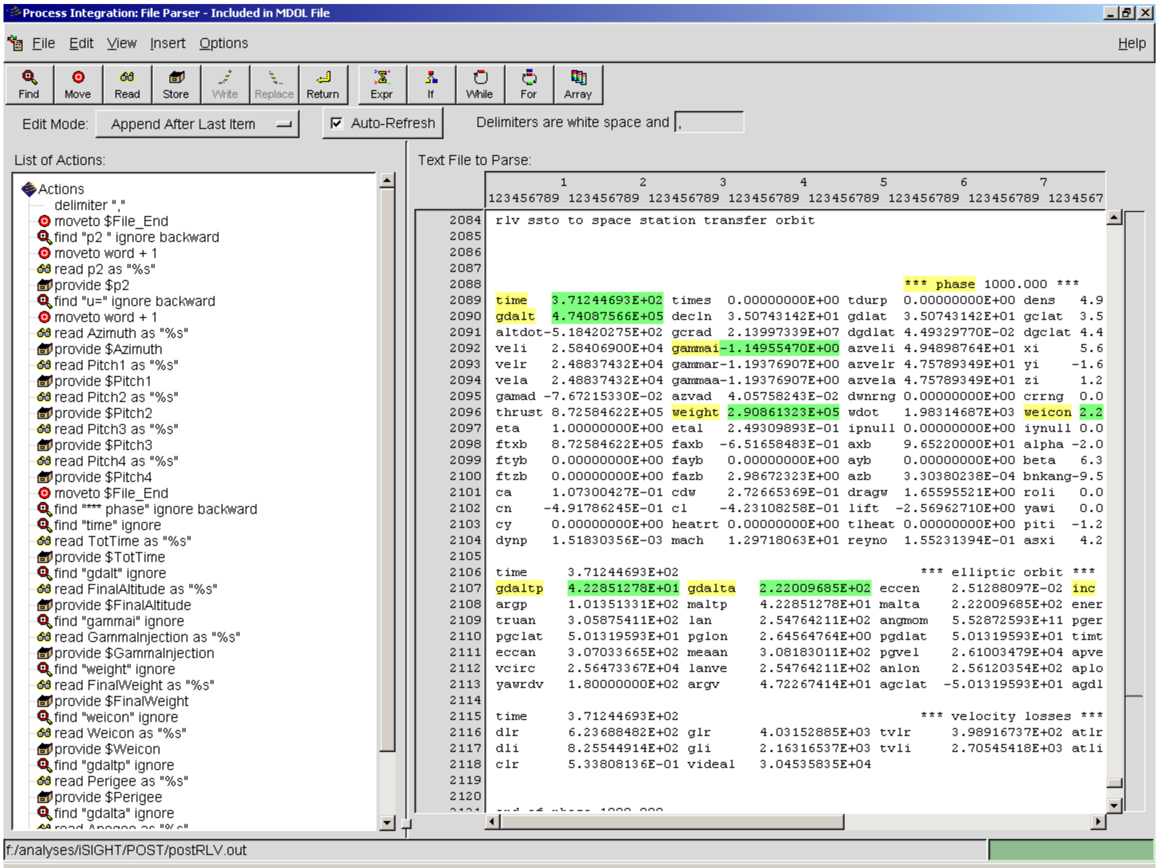
Overall, ModelCenter is very user-friendly. Most of the operations are performed with simple mouse commands and are usually very intuitive. A very good tutorial is provided with ModelCenter that walks the user through nearly all the capabilities of ModelCenter and the Analysis Server. From experience and observation, ModelCenter takes only a few days to pick up the basics and a week or two to master. Documentation is online, and is very good with several examples to illustrate various points. At the same time, the documentation is not so cumbersome that information becomes difficult to find. Technical support can also be obtained from ModelCenter directly via email. Responses are usually very helpful and take about a day or two. ModelCenter is currently being evaluated and used within the Intelligent Synthesis Environment (ISE) and the Advanced Concepts projects of NASA. Boeing and Lockheed Martin (Denver) have also adopted ModelCenter in their aerospace design activities. Finally, as the SSDL is part of the academic community, Phoenix Integration has agreed to provide licenses for ModelCenter and the Analysis Server free of charge.

iSIGHT

iSIGHT's origins date back to the late 70's and early 80's when Dr. Siu Tong worked on the concept of a "software robot" to perform the manual iteration typically performed by engineers in the design process. This research was the basis for Dr. Tong's dissertation at MIT. This software was further developed, under the name "Engineous," over an 11-year span during Dr. Tong's employment with the General Electric Company (GE). Shortly after Dr. Tong left GE, he founded Engineous Software in 1996 and the product was renamed iSIGHT. Dr. Tong continues to oversee the development iSIGHT as Engineous Software's board chair. Engineous Software is based out of North Carolina and has offices worldwide. iSIGHT is developed for Windows NT and Unix-based platforms. iSIGHT provides no support for Macintosh computers.

iSIGHT is based on its custom Multidisciplinary Optimization Language (MDOL). Though this language is not too difficult to learn and use, the GUI's within the iSIGHT application usually allow the user to avoid direct interaction with this language for simple tasks. MDOL is based on the use of what is referred to as "blocks". There are great many different types of blocks, each of which handles specific operations. Such operations include the overall control flow of the design problem, analysis code wrapping, simple internal calculations and system-level analysis methods. Within the iSIGHT GUI, these blocks are graphically represented, and the properties of each can modify by simply clicking on them.

The wrapping capabilities for command-line tools are quite good. The options available when generating and parsing input and output files are quite extensive. iSIGHT's capabilities in this area make dealing with complicated files slightly easier than with ModelCenter. The File Parser portion of iSIGHT (see Figure 11) provides the user with a GUI that allows him to easily define exactly how to search for and identify particular variables. Simple instructions, such as searching backwards through a file, are more straightforward in iSIGHT than in ModelCenter, and the interface allows the user to see the results of his parsing commands as he writes them. iSIGHT also provides an interface to Excel-based applications, allowing the user to specify inputs, define macros that must be run for the analysis, and which outputs to retrieve. Here, iSIGHT's capabilities are not as extensive as ModelCenter's but certainly adequate. iSIGHT also provides integration with MSC/NASTRAN, a commercial finite analysis code, but this feature was not reviewed.



Edit Mode: Append After Last Item

☑ Auto-Refresh

Delimiters are white space and ,

List of Actions:

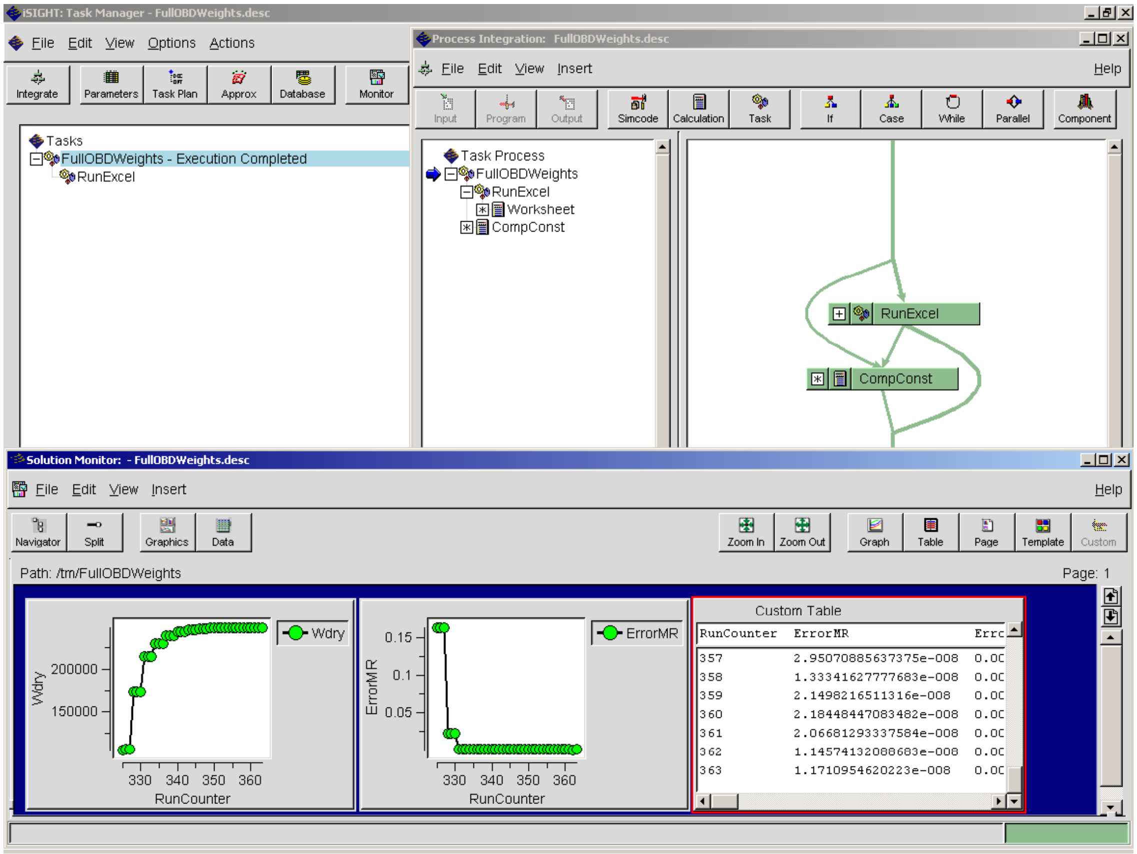
```
Actions
    delimiter ","
    moveto $File_End
    find "p2 " ignore backward
    moveto word + 1
    read p2 as "%s"
    provide $p2
    find "u=" ignore backward
    moveto word + 1
    read Azimuth as "%s"
    provide $Azimuth
    read Pitch1 as "%s"
    provide $Pitch1
    read Pitch2 as "%s"
    provide $Pitch2
    read Pitch3 as "%s"
    provide $Pitch3
    read Pitch4 as "%s"
    provide $Pitch4
    moveto $File_End
    find "*** phase" ignore backward
    find "time" ignore
    read TotTime as "%s"
    provide $TotTime
    find "gdalt" ignore
    read FinalAltitude as "%s"
    provide $FinalAltitude
    find "gammai" ignore
    read GammaInjection as "%s"
    provide $GammaInjection
    find "weight" ignore
    read FinalWeight as "%s"
    provide $FinalWeight
    find "weicon" ignore
    read Weicon as "%s"
    provide $Weicon
    find "gdaltp" ignore
    read Perigee as "%s"
    provide $Perigee
    find "gdalta" ignore
    read Apogee as "%s"
```

```
         1         2         3         4         5         6         7
123456789 123456789 123456789 123456789 123456789 123456789 123456789 1234567
rlv ssto to space station transfer orbit



                                                   *** phase 1000.000 ***
time    3.71244693E+02 times   0.00000000E+00 tdurp  0.00000000E+00 dens   4.9
gdalt   4.74087566E+05 decln   3.50743142E+01 gdlat  3.50743142E+01 gclat  3.5
altdot-5.18420275E+02 gcrad   2.13997339E+07 dgdlat 4.49329770E-02 dgclat 4.4
veli    2.58406900E+04 gammai-1.14955470E+00 azveli 4.94898764E+01 xi     5.6
velr    2.48837432E+04 gammar-1.19376907E+00 azvelr 4.75789349E+01 yi    -1.6
vela    2.48837432E+04 gammaa-1.19376907E+00 azvela 4.75789349E+01 zi     1.2
gamad  -7.67215330E-02 azvad   4.05758243E-02 dwnrng 0.00000000E+00 crrng  0.0
thrust 8.72584622E+05 weight  2.90861323E+05 wdot   1.98314687E+03 weicon 2.2
eta     1.00000000E+00 etal    2.49309893E-01 ipnull 0.00000000E+00 iynull 0.0
ftxb    8.72584622E+05 faxb   -6.51658483E-01 axb    9.65220000E+01 alpha -2.0
ftyb    0.00000000E+00 fayb    0.00000000E+00 ayb    0.00000000E+00 beta   6.3
ftzb    0.00000000E+00 fazb    2.98672323E+00 azb    3.30380238E-04 bnkang-9.5
ca      1.07300427E-01 cdw     2.72665369E-01 dragw  1.65595521E+00 roli   0.0
cn     -4.91786245E-01 cl     -4.23108258E-01 lift  -2.56962710E+00 yawi   0.0
cy      0.00000000E+00 heatrt 0.00000000E+00 tlheat 0.00000000E+00 piti  -1.2
dynp    1.51830356E-03 mach    1.29718063E+01 reyno  1.55231394E-01 asxi   4.2

time     3.71244693E+02                               *** elliptic orbit ***
gdaltp   4.22851278E+01 gdalta   2.22009685E+02 eccen    2.51288097E-02 inc
argp     1.01351331E+02 maltp    4.22851278E+01 malta    2.22009685E+02 ener
truan    3.05875411E+02 lan      2.54764211E+02 angmom   5.52872593E+11 pger
pgclat   5.01319593E+01 pglon    2.64564764E+00 pgdlat   5.01319593E+01 timt
eccan    3.07033665E+02 meaan    3.08183011E+02 pgvel    2.61003479E+04 apve
vcirc    2.56473367E+04 lanve    2.54764211E+02 anlon    2.56120354E+02 aplo
yawrdv   1.80000000E+02 argv     4.72267414E+01 agclat  -5.01319593E+01 agdl

time     3.71244693E+02                               *** velocity losses ***
dlr      6.23688482E+02 glr      4.03152885E+03 tvlr     3.98916737E+02 atlr
dli      8.25544914E+02 gli      2.16316537E+03 tvli     2.70545418E+03 atli
clr      5.33808136E-01 videal   3.04535835E+04
```

f:/analyses/iSIGHT/POST/postRLV.out

Figure 11: File parsing in iSIGHT

Once an analysis tool is wrapped, iSIGHT does possess the ability to save this configuration for inclusion into other design problems, but there are some limitations to its reusability. In iSIGHT, links between various analyses are not explicitly defined. If a variable is to be coupled between two analyses, it must be similarly named in each. Data flow is defined by specifying the order in which the analyses are to be run. Therefore, when reusing wrapped analyses, variable names might need to be modified to ensure proper integration with the rest of the model. However, at the same time, this strategy simplifies problem definition as links between disciplines do not need to be explicitly specified. Also, the need to have identical variables defined multiple times within different analyses is eliminated.


iSIGHT: Task Manager - FullOBDWeights.desc
File Edit View Options Actions
Integrate Parameters Task Plan Approx Database Monitor
Tasks
FullOBDWeights - Execution Completed
RunExcel
Process Integration: FullOBDWeights.desc
File Edit View Insert Help
Input Program Output Simcode Calculation Task If Case While Parallel Component
Task Process
FullOBDWeights
RunExcel
Worksheet
CompConst
RunExcel
CompConst
Solution Monitor: - FullOBDWeights.desc
File Edit View Insert Help
Navigator Split Graphics Data Zoom In Zoom Out Graph Table Page Template Custom
Path: /tm/FullOBDWeights
Page: 1
Wdry
200000
150000
330 340 350 360
RunCounter
Wdry
ErrorMR
0.15
0.1
0.05
330 340 350 360
RunCounter
ErrorMR
Custom Table

RunCounter	ErrorMR	Errc
357	2.95070885637375e-008	0.0C
358	1.33341627777683e-008	0.0C
359	2.1498216511316e-008	0.0C
360	2.18448447083482e-008	0.0C
361	2.06681293337584e-008	0.0C
362	1.14574132088683e-008	0.0C
363	1.1710954620223e-008	0.0C

Figure 12: Optimization and Problem Setup in iSIGHT

When setting up the design structure matrix and data flow of a problem, iSIGHT's capabilities are thorough and well implemented. The ability to include logic based control flow, such as while-loops, if and case statements is built in with very easy-to-use graphical interfaces for each. Unlike in ModelCenter, the user does not need to create a custom scripts to create this kind of control; it is built in. Additionally, iSIGHT also provides the ability to specify parallel branches in data flow, and allow analysis codes to actually run in parallel. This capability is greatly advantageous when working with complex design problems that use optimization strategies designed to take advantage of this. The parallel execution of analyses also extends to the ability to run several instance of the same analysis at once, enabling more than one run within a DOE to be computed at the same time.

iSIGHT's major limitations come in its ability to easily distribute the execution of analyses to other machines, especially across platform boundaries. It is fairly easy to set up an Excel-based analyses to run on the same NT machine that iSIGHT runs. It is similarly easy to integrate command-line analyses that run on Unix machines, when also running iSIGHT on a Unix platform. However, when a problem requires that analyses cross platform boundaries, integration becomes fairly difficult. The wrapping instructions for a given analysis are maintained in the problem definition at the system level. Therefore, whenever one wants to distribute an analysis to another machine, he must share a common directory structure with the machine running iSIGHT. This is usually not a problem when sticking to either Unix or Windows machines, but become quite tedious when using both. Additionally, the support for Excel-based analyses is limited to allowing their execution on the same NT machine on which iSIGHT is running. Needless to say, the advantage of being able use parallel execution is greatly reduced when several of those analyses must be run on the same machine. With regards to keeping disciplinary experts "in-the-loop," iSIGHT does not provided any means prompt disciplinary experts for verification of analysis results. However, iSIGHT does provide extensive capabilities for the expert to set up several knowledge-based rules that govern how an analysis will run, and what additional instructions must conditionally be performed. Though this strategy is not as good as allowing the disciplinary expert to directly monitor and verify results during system level analysis, it is better than nothing.

iSIGHT's greatest strengths lie in the area of system-level analysis and optimization. iSIGHT includes a wide range of optimization methods, including several gradient-based algorithms, genetic algorithms, simulated annealing, and methods which can accommodate discrete and integer variables. These methods are listed in Table 5. Access is made available to all the internal parameters of these methods, and documentation is provided on how each of them work. Directly supplying gradient information is possible for those methods that can use it. Doing so does require the

Table 5: Optimization Algorithms available in iSIGHT

Name
Exterior Penalty -ADS
Modified Method of Feasible Directions -ADS
Sequential Linear Programming -ADS
Method of Feasible Directions -CONMIN
Mixed Integer Optimization - MOST
Sequential Quadratic Programming - DONLP
Sequential Quadratic Programming - NLPQL
Successive Approximation Method
Genetic Algorithm
Simulated Annealing
Directed Heuristic Search

user to have a more solid understanding of the MDOL, it is not too difficult to implement. iSIGHT also provides the ability to run Monte Carlo simulations.

The support for Design of Experiments and Response Surface approximations is extremely thorough. Several options are available when determining the array of experiments required for a response surface formulation. Table 6 summarizes the general categories of DOE techniques available in iSIGHT. For each option in the table, there are several more options available that allow the user further refine what kind of study is performed. For example, when selecting Orthogonal Arrays, it appeared that nearly all of the well-used Taguchi arrays were available. The ability for the user to import custom DOE runs is also available. The capability to specify noise variables and produce ANOVA tables is included. Additionally, there is added support for creating and using response surface approximations to replace the actual analysis of extremely computationally intensive codes. While iSIGHT does provide the ability to integrate user supplied optimization packages, the variety of optimization methods included is so extensive, it is doubtful the SSDL's needs would ever require this option to be used; consequentially this capability was not extensively examined. Another aspect of iSIGHT's system-level analysis that was found to be particularly innovative was the automatic inclusion of particular constraint variable in all optimizations. This variable indicates whether an analysis ran properly or not. Using this special constraint, if during an optimization, a configuration is chosen that causes one of the disciplinary analyses to fail, rather than halt the optimization process, that point design is heavily constrained. This allows the optimization to continue while directing the optimizer away from design points that cause problems for the analytical tools.

Table 6: Design of Experiments options in iSIGHT

Name
Full Factorial Design
Orthogonal Arrays (Taguchi)
Latin Hypercubes
Central Composite Design

iSIGHT also includes a very detailed and flexible means for monitoring/reviewing the results. This capability allows the user to create custom tables containing only the variables, constraints and objectives of interest. The graphing capabilities of iSIGHT provide a wide variety of options, nearing the capabilities typically found in programs like Microsoft's Excel. One also has the option to define several different page layouts, which can display various combinations of plots and tabulated results. Results can be easily stored and reviewed, however, the ability to reset the current values of all design variables to a previously saved point, does not seem to be available.

In general, iSIGHT is not as user-friendly as ModelCenter, but this difference is usually most prevalent when using capabilities that are not available in ModelCenter at all. Because, the overall capabilities of iSIGHT are more extensive, it is a more difficult to pick up and master, but certainly not at an intolerable level. The documentation provided with iSIGHT is extremely thorough. At times however, it seems that the large amount of detailed information can become somewhat overwhelming when trying to find the answer to a specific question. However, printed versions of the documentation were provided in addition to the electronic versions included with the software. Though examples are provided, they are not necessarily presented in any easy to follow, walkthrough manner. A single tutorial is available that teaches the user how to wrap a simple code, set up an optimization, and review the results. Unfortunately, this tutorial by no means familiarizes the user with all the capabilities available in iSIGHT, and in fact leaves much to be desired. Additional tutorials detailing how to wrap Excel-based tools and distribute analysis to other computers would be extremely helpful. Customer support is available by phone and email, was quite helpful, and usually responded within a day or less.

While iSIGHT's capabilities are much better in many areas, its limitations are in key areas of importance for the SSDL. These limitations became most apparent in the attempt to implement the sample RLV problem in iSIGHT. It was fairly easy to integrate POST and SCORES into a common iSIGHT model, as they both run on a UNIX platform. Similarly, it was not very difficult to integrate the Excel-based GT-Sizer code into an iSIGHT Model running on the same Windows NT machine. However, attempts to integrate all the codes required for the sample RLV problem into one common model were unsuccessful. This was due mainly to the limitations of iSIGHT's abilities to integrated codes across platform boundaries. The parallel execution and optimization capabilities were successfully tested using just SCORES and POST together, but it was not possible to do so with all the codes required for sample RLV problem.

iSIGHT is well used in a variety of engineering industries, including aerospace. NASA’s ISE program is also looking at iSIGHT in its research and applications. iSIGHT is naturally used by GE, with which it shares a strong partnership. iSIGHT is also involved a new research program, RaDEO (Rapid Design Exploration and Optimization). This program will unite the efforts of several organizations including NASA and Lockhead Martin. As with Phoenix Integration, Engineous Software has graciously provided several licenses of iSIGHT to the SSDL free of charge.

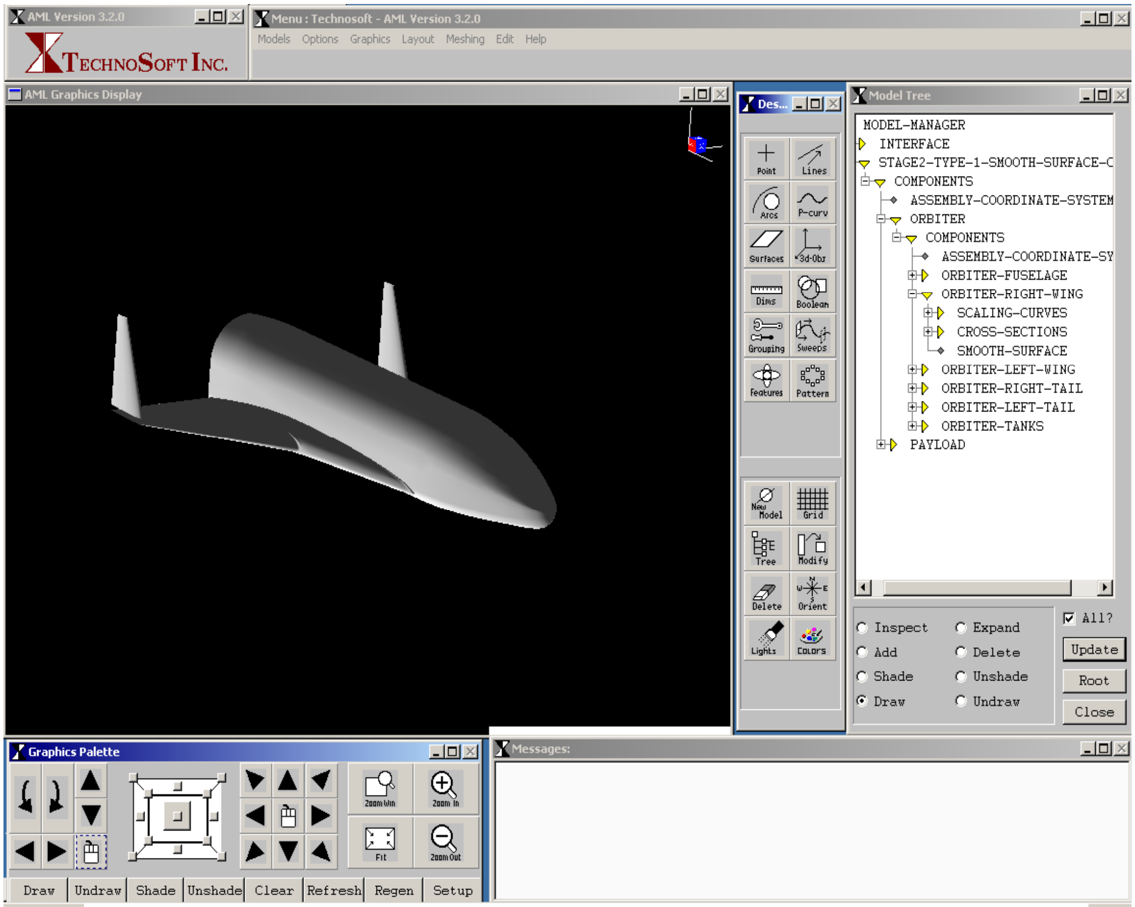
AML

The Adaptive Modeling Language (AML) is the main product of Technosoft Inc. Based out of Cincinnati Ohio, Technsoft is headed by Dr. Adel Chemaly, who took his technical expertise from academia to start the company which now has clients and offices in the U.S., Mexico, and Europe. As President, Adel Chemaly oversees the development of AML, which is available for both Unix-based systems and Windows 95/98/NT. Macintosh support for AML is not available.

```
(define-class TABLE
:inherit-from (assembly-object)
:properties (
table-height 36.0
table-width72.0
table-depth36.0
thickness  1.0
total-weight (loop for i in (select-object
:from (the table)
:class ‘material-object
:eval ‘(the weight))
sum i)

          object-list(select-object
:from (the table)
:class ‘graphic-object
:test ‘(not (eq (the) ,(the table))))

          display?nil
)
:subobjects (
(origin :class ‘point-object
coordinates ‘(0.0 0.0 0.0)
display?t
)

(top :class ‘table-top
height^^thickness
width^^table-width
depth^^table-depth
display?(not (the table display>))
reference-object(the table)
orientation(list (translate (list 0.0
(- (the table-height)
(half (the thickness)))
0.0)))
)
(legs :class ‘series-object
series-prefix‘leg
class-expression‘table-leg
init-from
‘(display?(not (the table display?))
  height(- !table-height !thickness)
diameter!thickness
reference-object(the table)
orientation
(list
(rotate 90.0 :x-axis)
(translate
(list (if (evenp (the index))
(-  (half (the table-width))
(the thickness))
(-(the thickness)
(half (the table-width)))
)
(half (the height))
(if (> (the index) 1)
(- (half (the table-depth))
(the thickness))
(- (the thickness)
(half (the table-depth)))
          ))))
)
quantity4
)
)
```

Figure 13: Sample AML code

At its root, AML is exactly what its name describes, a modeling language. The most basic use of AML is through the declaration of classes, which inherit from AML-developed primitives. AML is built on the solid philosophy of object-oriented software design, inheriting all the benefits and advantage therein. Figure 13 shows an example of a table model implemented in AML. To effectively create models in AML, the user must have good understanding of computer programming practices, in particular those of object-oriented programming. AML is based on the LISP programming language, which has the disadvantage that is fairly uncommon, requiring the most users to familiarize themselves with the language before being becoming a proficient user of AML. Though the language is not particularly any more difficult to learn than other object-oriented languages, such a C++, is does have a few little quirks, like the use of reverse Polish notation for all mathematical notations.



```
MODEL-MANAGER
 INTERFACE
 STAGE2-TYPE-1-SMOOTH-SURFACE-C
   COMPONENTS
     ASSEMBLY-COORDINATE-SYSTEM
     ORBITER
       COMPONENTS
         ASSEMBLY-COORDINATE-SY
         ORBITER-FUSELAGE
         ORBITER-RIGHT-WING
           SCALING-CURVES
           CROSS-SECTIONS
           SMOOTH-SURFACE
         ORBITER-LEFT-WING
         ORBITER-RIGHT-TAIL
         ORBITER-LEFT-TAIL
         ORBITER-TANKS
     PAYLOAD
```

○ Inspect ○ Expand ☑ All?
○ Add ○ Delete
○ Shade ○ Unshade
● Draw ○ Undraw

Figure 14: Geometric modeling in AML

One of the key advantages of AML lies in its geometric modeling capabilities. AML provides a very extensive suite of classes, functions and methods that allow the user to parametrically build up nearly any kind of geometry. All geometric primitives are defined by a few specific parameters, allowing the user to build more complex geometries that depend on any number of real-world design variables. In testing, the

]

geometry of simple RLV was generated in this fashion (see Figure 14). Key variables that defined the geometry included parameters such as fuselage length and diameter, aspect ratio and tail orientation angle, to name a few. By changing the value of these variables, the entire geometry of vehicle is automatically regenerated in moments. AML does provide built-in graphical user interfaces to aid in the creation and inspection of geometries, but complicated models invariably require direct coding in AML. As much of AML's key functionality involves geometric modeling, it does not need nor have any integration with other external CAD packages. AML's geometric modeling capabilities and user-friendliness come nowhere near those of traditional CAD packages. However, having this capability built into the framework provides the unique advantage of allowing the geometry and any analysis that depends on it, to always be up-to-date as a system-level optimizer or other discipline change it.

Of particular interest to the SSDL would be AML's integration with the SHABP portion of the aerodynamic analysis code APAS. Unfortunately, Technosoft's efforts to fully integrate the remaining portions of APAS, such as UDP, are still in progress. Once a geometry has been created in the AML framework, classes are available which will generate a mesh of that geometry, send that data to SHABP, and generate POST aerodecks fairly seamlessly. Once set up, the geometry of the model can easily be changed and new aerodynamic data obtained in minutes. However, unless several aerodynamic analyses plan to be conducted, the extra time required to create a geometric model in AML might not be worth it. Integrating codes, like SHABP, into AML requires a great deal of development and is usually performed by AML employees specifically for their clients. Unfortunately, AML's strategy towards analyses integration seems more inclined towards this type of sophisticated one-time integration with key, widely-used analytical tools. Though this strategy produces very impressive results for the particular analyses AML integrates, it is not the kind of flexible integration that is would be suited for a conceptual design organization. This is especially true for the SSDL, as its members are constantly developing new tools and changing existing ones.

Variables are defined in AML by creating instances of appropriate primitive classes. Though AML includes menu driven and graphical interfaces to aid in the creation of a design model, even the simple declaration of new variables requires some programming know-how. When using formulas to define variable values, AML keeps track of which variables depend on others, and ensures that when any value is demanded, the appropriate calculations and analyses on which the value depends, are run. However, simple formulas do require the use of reverse Polish notation, which does take some getting used to.

AML claims to support the ability to integrate command-line and Excel-based analysis codes, however these capabilities, which are key for the SSDL, are not included in the standard release. Unfortunately, it was not possible to obtain the additional classes that enable this functionality from Technosoft; consequentially, this feature could not be evaluated. Without the key ability to integrate Excel-based and command-line tools, other requirements, such as the ability to distribute these analyses could not be evaluated. Technosoft claims to have integrated various optimization methods with AML, including Powell's method, a genetic algorithm, and what they referred to as MDO. However, efforts to acquire the additional software required to test this functionality were similarly unsuccessful.

Overall, AML is not very user-friendly to new users. Any user of AML should have a good understanding of basic programming practices, particularly those of object-oriented design. AML takes a few weeks to learn and several to master. Technosoft does, however, have very good training manuals that walk the user through the various elements and operations of AML. Documentation is very good, providing clear and thorough explanation of how all the classes, methods and functions in AML work. Customer support is good, being accessible via either email or phone. It is known that AML typically works very closely with its customers in developing sophisticated analysis models for their specific design problems.

As it was not possible to obtain the classes required to integrate Excel-based or command-line codes, implementation of the sample RLV problem in AML was not possible. It was possible to parametrically model the RLV's geometry, but with this capability alone, no system level design optimizations or analyses could be performed.

AML is currently the key partner in a design study with the Air Force, Lockhead Martin and the SSDL. The Supersonic / Hypersonic Vehicle Design (SHVD) program plans to integrate an impressive list of industry standard analysis codes for space vehicle design within the AML framework. As this project has yet to really begin, no information is available on how well this implementation is progressing. However, if this project is successful it should leave AML with a great list of high fidelity analysis capabilities. AML might make a good choice for design organizations that require the use of many high fidelity, difficult to wrap codes and are willing and able to invest the time and money to integrate those codes with AML. However, the time required to learn AML and its lack of flexibility make it an impractical choice for an organization which primarily does conceptual to preliminary level design such as the SSDL. Technosoft has graciously supplied the SSDL with copies of AML free of charge.

SUMMARY

In conclusion, each the three frameworks all seem to have their strengths and weaknesses in different areas. The ideal choice would most likely combine ModelCenter's user friendliness and flexibility in wrapping and distributing disciplinary analyses, iSIGHT's ability to run those analyses in parallel in conjunction with its extensive system-level analysis capabilities, and AML's integrated parametric modeling capabilities. Sadly, such a product does not yet exist. One of ModelCenter's key drawbacks is its limited optimization capabilities, which do not include stochastic methods nor support for discrete variables. ModelCenter also lacks the important ability to perform analysis in parallel and built-in capabilities for Monte Carlo simulations. iSIGHT, is slightly more difficult to use and its leaves much to be desired about its ability to easily distribute analysis to multiple computers across platform boundaries. AML, takes quite a bit of time, and preferably training, to learn and seems more oriented towards one-time integration solutions for analysis tools.

Though none of the candidate packages currently contain all the features that the SSDL is looking for broad lab-level, ModelCenter seems to satisfy most of the key requirements. ModelCenter is easy to use and learn. Already the members of the SSDL are successfully using ModelCenter in two major projects with very promising results. Additionally, Phoenix Integration has said that the ability to run analyses in parallel will be address in the next release sometime late in the year 2001.

iSIGHT is close second as its potential capabilities are much greater. Supposedly, the major limitations noticed about iSIGHT are to be resolved in the release of version 6.0, due at the end of April 2001. However, even with potential improvements, iSIGHT does take a bit longer to learn, and is not nearly as intuitive to use as ModelCenter. It is likely though, that the SSDL will probably continue to use iSIGHT as there are there are several specific types of system-level analysis and optimization approaches that only it has the built-in ability to perform.

As previously mentioned, AML does not fit the needs of the SSDL well. Technosoft's approach to design solutions seems to be more oriented towards the development of tightly integrated applications tailored to the specific kind of high fidelity, configuration specific design that its customers require. The SSDL would like to adopt a framework package that is flexible and easy to adapt to a wide range of problem in a short time. ModelCenter bests suits this requirement as the time needed to set-up and debug the analysis of new problems from scratch takes only about a week. However, the SSDL will naturally continue to consider developments of these and other integrated design framework solutions in the future.

REFERENCES

1. Acton, D.E., Olds, J. R., “Computational Frameworks for Collaborative Multidisciplinary Design of Complex Systems” AIAA 98-4942, 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, September 2-4, 1998.

2. Cormier, T., Scott, A., Ledsinger, L., McCormick, D., Way, D. W., Olds, J. R., “Comparison of Collaborative Optimization to Conventional Design Techniques for a Conceptual RLV,” AIAA 2000-4885, 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, September 6-8, 2000.

3. Phoenix Integration, URL: http://www.phoenix-int.com

4. Engineous Software, URL: http://www.engineous.com

5. Technosoft Inc., URL: http://www.technosoft.com