8-2004

Interactive Design Using CFD and Virtual Engineering

Gengxun Huang
Iowa State University

Kenneth M. Bryden
Iowa State University, kmbryden@iastate.edu

Douglas S. McCorkle
Iowa State University, doug@agsolver.com

Interactive Design Using CFD and Virtual Engineering

Abstract

Virtual engineering is a powerful concept, defined as a technology that integrates geometric models and related engineering tools such as analysis and simulation, optimization and decision-making tools, etc. within a computer generated environment that facilitates multidisciplinary and collaborative product realization [1]. Virtual engineering applications can be constructed from scratch with high-level programming languages. However, since the end-user of the virtual engineering application is most likely not a programming expert, high-level support is needed to provide the user with the capability to construct his own application in an intuitive manner and with minimal coding. In this paper, we present a framework of the virtual engineering environment and its implementation, identify the general requirements for a virtual engineering application, and summarize the architecture. A virtual engineering application on computational fluid dynamics (CFD)-based interactive design is used to motivate the research as well as to evaluate the performance of the system. The sample application is related to the coal transport system of a coal-fired power plant. Finally, the topics for future research are given.

Disciplines

Computer-Aided Engineering and Design

Comments

This paper is from *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA 2004-4364, Albany, NY, August 2004.

10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference
30 August - 1 September 2004, Albany, New York

AIAA 2004-4364

Interactive Design Using CFD and Virtual Engineering

Gengxun Huang, M.S,[*] Kenneth M. Bryden, Ph.D.[†]
Mechanical Engineering Department, Iowa State University, Ames, IA, 50010

Douglas S. McCorkle, M.S.[‡]
Mechanical Engineering Department, Iowa State University, Ames, IA, 50010,USA

Virtual engineering is a powerful concept, defined as *a technology that integrates geometric models and related engineering tools such as analysis and simulation, optimization and decision-making tools, etc. within a computer generated environment that facilitates multidisciplinary and collaborative product realization* [1]. Virtual engineering applications can be constructed from scratch with high-level programming languages. However, since the end-user of the virtual engineering application is most likely not a programming expert, high-level support is needed to provide the user with the capability to construct his own application in an intuitive manner and with minimal coding. In this paper, we present a framework of the virtual engineering environment and its implementation, identify the general requirements for a virtual engineering application, and summarize the architecture. A virtual engineering application on computational fluid dynamics (CFD)-based interactive design is used to motivate the research as well as to evaluate the performance of the system. The sample application is related to the coal transport system of a coal-fired power plant. Finally, the topics for future research are given.

Nomenclature

D	=	objective function	θ	=	orientation of the orifice
S	=	constraint function	$\tilde{r}$	=	ratio of the radius of orifice and pipe
$\vec{p}$	=	vector of design parameters	$S_{orifice}$	=	area of the cross-section of orifice
$\vec{p}^*$	=	vector of optimal design parameters	S_{pipe}	=	area of the cross-section of pipe
$\vec{f}$	=	vector of flow field variables	v_k	=	gas velocity magnitude at the exit
F	=	flow governing equations			
B	=	flow boundary equations			
K	=	number of the cells at the exit surface of the pipe			

I. Introduction

Virtual engineering is an emerging technology is becoming popular and is defined as *a technology that integrates geometric models and related engineering tools such as analysis and simulation, optimization and decision making tools, etc. within a computer generated environment that facilitates multidisciplinary and collaborative product realization* [1]. Figure 1 illustrates the basic functionality of a virtual engineering system. The ultimate goal of virtual engineering is to provide a user-centered, first-person perspective that enables engineers to

[*] Graduate Research Assistant, Department of Mechanical Engineering, Email: hgx@iastate.edu AIAA Student Member
[†] Corresponding Author, Associate Professor, ASME Member.
3030 Black Engineering Bldg, Iowa State University, Ames, IA50011
Email: kmbryden@iastate.edu, Phone/Fax: 515-294-3891/3261.
[‡] Graduate Research Assistant, Department of Mechanical Engineering, Email: mccdo@iastate.edu AIAA student Member

interact with the designed product in a natural way and to provide the engineers with a wide range of accessible engineering tools. Although the concept of virtual engineering is new, several research groups have worked to couple engineering analysis and virtual environments, enabling the user to perform engineering tasks from the virtual environment [2, 3, 4, 5]. One of the earliest applications was Boilermaker [2]. Boilermaker coupled a computational model of an industrial furnace with a virtual environment to enable the user to make changes to the inlet nozzle configuration and then see the resulting furnace performance on the fly. This enabled the engineer to explore various furnace configurations on the fly and consider a number of variables to create the optimum design for a particular furnace. Another example of coupling engineering analysis and virtual environments is DN-Edit [4]. DN-Edit allows NURBS-based (Non-Uniform Rational B-Splines) surface geometry to be altered interactively. Virtual cursors allow for interaction with geometry surfaces and enable surface points to be displaced and materials to be added or removed and provide exact control of surface normals at specific points. The resulting surfaces are NURBS-based and can be exported to various CAD or analysis programs. The shaping of three-dimensional geometry in a virtual environment allows for an intuitive process, bypassing the obstacles and limitations of a two-dimensional human-computer interface. Additionally, applications coupling virtual environments and engineering have been developed for assembly [3], hose routing, and design of three-dimensional mechanisms [5].

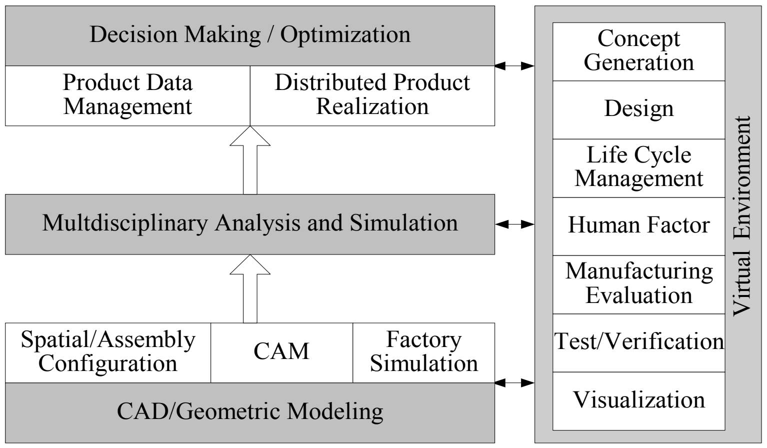


Figure 1 Framework of the Virtual Engineering System [1]

According to the definition of virtual engineering, the key components of a virtual engineering application include: an engineering analysis engine that simulates the behavior of the products; a dynamic visualization environment that allows users to control the computational engine and visualize the simulation results in real time; decision support tools providing suggestions for the product realization problems; and a user interface module expanding the scope of virtual reality (VR) applications from visualization to interaction. From a software engineering perspective, the engineering analysis engine and decision support tools can be integrated into one computational engine. Virtual engineering applications can be constructed from scratch with high-level programming languages. However, as with any construction task, applications can be constructed more easily and efficiently when higher level support is provided. The general virtual engineering environments that contain software tools should be able to provide higher level functionality on aspects common to arbitrary virtual engineering applications. Existing research on coupling engineering analysis and the virtual environment, discussed above, shows that the construction of the virtual engineering application is an underdeveloped aspect; existing systems can only support specific applications. In addition, existing systems are not tailored to the specific requirements of an effective virtual engineering application. This led to the main research goal of this paper: the development of concepts and techniques for a general purpose virtual engineering environment that fills the gap in existing systems. The motivation and test bed for this system come from a computational fluid dynamics (CFD)-based interactive design tool using virtual engineering concepts to design a coal transport pipe.

The paper is organized in the following manner: Section II describes the basic requirements for virtual engineering applications; interactive design system is briefly discussed in Section III; Section IV presents the details of the software architecture and its implementation; and Section V provides an example of using interactive design in coal pipe design. Finally, Section VI provides conclusions, discussion, and areas for future work.

II. General Requirements of Virtual Engineering Applications

Virtual engineering is a powerful concept; using this technology enables the user to walk through the operating system and observe how it works, and how it responds to changes in design, operation, or any other engineering modification [1]. Evidence shows that virtual engineering can greatly enhance productivity. For example, when combined with virtual engineering technology, the time between changes to parameters and the visualization of the results in the interactive design process can be dramatically reduced, allowing the user more time to perform what-if analysis.

However, great power comes with great responsibility. From the software perspective, four basic components comprise a VE system: the integrated application, the user interface, the visualization environment, and of course, the communication and data transfer among the previous three components. Each of these components has to be considered when developing a virtual engineering application. The choice of the architecture for the application is determined by the requirements for the application integration and data communication components. Requirements are formulated and possible solutions are discussed.

First, a crucial aspect of a general virtual engineering system is that the existing engineering applications should be easy to transform into virtual engineering applications. Most current processes for engineering applications for systems, architectural and engineering designs, heat and mass constitutive simulations, cost estimations, and synthetic environment visualizations are self-contained, and each provides its own user interface. Obviously, it is not practical to replicate the powerful in-house or commercial software packages from scratch. Therefore, although integration of engineering tools within a virtual engineering system can provide a powerful cost-reduction tool, it can also be very difficult. In order to achieve acceptable interaction and flexibility, the software architecture cannot be coupled with a single software package. Thus, a general method for easily integrating existing engineering tools into the virtual engineering system must first be implemented. A feasible approach to program instrumentation is to break the code up into various modules that the user controls explicitly by issuing a sequence of commands. One popular approach for doing this is to glue separate components together with scripting languages such as Python, Ruby, etc. The advantage of this method is that it maximizes reuse of the original scientific code, and the code base is easy to modify and maintain, as scripting languages are more user friendly than industrial strength programming languages such as C and C++. This makes it suitable for controlling most applications, including large parallel applications on both distributed and shared memory systems [6].

Second, the engineering analysis severely limits interaction with the program during execution and makes visualization and interaction slow and cumbersome, especially if analysis needs to be carried out on a different system (such as a specialized graphics workstation for visualization). Some researchers use approximation methods or low fidelity models to construct surrogate mathematical models of the analysis results; thereafter, approximate results can be computed inexpensively. However, in some cases, high fidelity models have to be used, especially at the final stages of design optimization. The long computational time caused by high fidelity models makes real-time interaction very difficult; high fidelity models include using techniques such as CFD, finite elements analysis (FEA), and optimization of complex systems

Third, computational scientists have a wide variety of needs, and they may also want to write extensions to or integrate the system into existing applications. However, doing so requires a detailed understanding of every facet in the problem; because of this, many potential users ignore the power of virtual engineering. While there is no apparent solution to this problem, it is clear that the issues will need to be addressed in order for virtual engineering to be adapted into mainstream scientific computing. From the user's perspective, the basic requirement for the structure of this virtual engineering system is to make it very general and easy to add new capabilities so that users from different areas can extend the modules without being worried about the system's programming issues. We are in the process of developing such a virtual engineering design system.

Fourth, a key aim of virtual engineering is to engage the human capacity for complex system evaluation. Many researchers show that by creating a realistic experience from computational simulation and maintaining the visitor's focus within that experience, the maximum potential of human thought and skill can be utilized. This enables the user to focus entirely on the engineering problem, ensuring that complexities can be understood and that the full range of engineering solutions can be explored. However, many scientific computing problems involve complex simulation and generate high-dimensional datasets. The complexity of both the simulation and the generated data are too vast to analyze analytically or numerically. For example, in the case of interactive design, designers cannot decide which design is superior by simply looking at the parameters or data. On the other hand, the designer can observe the analysis result of a design and identify whether it is the result he/she wanted, given that the result is shown in a realistic and intuitive manner. Therefore, VR becomes an almost ideal technology for helping designers select the healthy initial population at the beginning of the optimization process. For these situations, immersive visualization and interaction is more appropriate for rapid inspection and manipulation of data. Bear in mind that the researcher who constructs an interface for a particular application is most likely not a visualization or computer graphics expert. Therefore, tedious coding on the visualization part should be kept to a minimum for the user.

Fifth, as mentioned before, in many cases, the surrounding environment is shown as a three-dimensional graphical environment. Interaction within the virtual environment should take place with an easily understood interface, appropriate to the user's technical background and expertise, thus enabling the user to explore and discover unexpected but critical details about the behavior of the system. Another requirement is intuitive input devices that can be used by a user to communicate with the environment. The most common are sensor devices that measure the 6 degrees of freedom (DOF) one has to move around in three-dimensional space (position and orientation). Because the traditional user interface in engineering applications is based on the desktop PC, the typical virtual environment interactive devices such as the wand are unsuitable for GUI interaction. However, using a virtual environment should not prevent the use of the traditional two-dimensional GUI.

Sixth, as noted earlier, engineering is more than analysis and design. A methodology for storage and rapid access to engineering analyses, plant data, geometry, and all other qualitative and quantitative engineering data related to plant operation needs to be developed.

The requirements discussed above are merely general requirements; it is up to the users to determine the special requirements for their individual applications. Here, we use the CFD-based interactive design as an example to explain how to implement the virtual engineering system.

III. Virtual Engineering Application: CFD-Based Interactive Design

3.1 Motivation

A shape optimization algorithm consists of an objective function and a state equation in addition to an optimization algorithm. The state equation is a discrete fluid flow model, i.e., a CFD model for the fluid flow problem in question. The objective function measures the quality of the design in a mathematical form. It depends on the solution of the CFD model, which depends on the shape of the flow domain. The flow domain or part of it can be described with the help of a finite number of design parameters, which are optimized. The optimization algorithm can, in principle, be any of several well-known methods: evolutionary algorithms (EAs) gradient method, conjugate-gradient method, etc. It is worth mentioning that the traditional CFD design optimization process as shown in Figure 2 is a two-cycle procedure. The optimization process starts by giving a proposed design $\vec{p}$. The inner iterative cycle solves the analysis problem for $\vec{f}$ given a proposed design $\vec{p}$, while the outer iterative cycle determines the optimum $\vec{p}$. Because the values of objective and constraint functions depend on the flow field solutions, the flow governing equation F must be solved every time D is evaluated. The system optimization continues until the optimization converges and an optimal system is obtained.

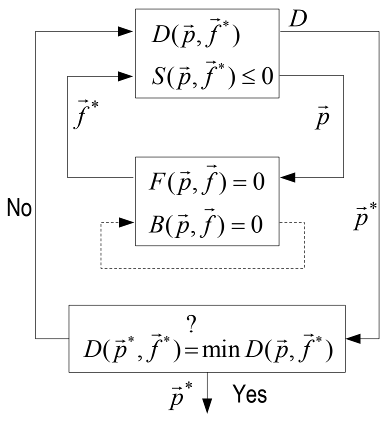


Figure 2 Traditional Two-Cycle Optimization Procedure

In terms of development cost of a new product, both efficiency and flexibility play an important role, but, unfortunately, high efficiency is often sacrificed for high flexibility in a design tool. For example, EAs are suitable tools for shape optimization, due to their flexibility, robustness, and ease in setting up a complex design problem. However, all these advantages related to the use of EAs have to be paid for with a high computational cost: a large number of objective function evaluations are needed to reach a satisfactory solution. Combining EAs with high fidelity CFD models further compounds the issues of computational cost associated with the design of thermal systems. It is not uncommon for one CFD analysis case to take weeks to finish. Therefore, it would take an unacceptably long time to solve a CFD-related design optimization problem. Consequently, thermal system design optimization is routinely performed using lower fidelity models or even approximation models, which cannot ensure the accuracy and quality of the optimization result for complex problems. Therefore, in some cases, high fidelity models have to be used, especially at the final stages of design optimization. For example, in the two-phase gas solid flow, the particle distribution can be calculated only with high fidelity models. Thus the general evolutionary algorithms, although offering great robustness and an unparalleled flexibility in setting up the design problem, do not show advantages when computational efficiency matters. Indeed, efficiency, in terms of minimization of computational effort, is a critical issue of thermal system shape design, and is one of the driving forces of technological developments in this field.

Basically, the challenge for research in CFD-based design optimization with evolutionary algorithms is the improvement of computational efficiency. There are three methods that can be employed to reduce the computation cost of a CFD design optimization problem:

(1) Reducing the number of calls to the solver of high fidelity CFD models. This requires the optimization method to be computed efficiently so that better designs can be found with fewer calls to the CFD solver.
(2) Speeding up the convergence rate when calculating the CFD model.
(3) Introducing the human experience into the optimization process to narrow the search space to a manageable size.

The first method will not be covered in this paper. The second method has been discussed in [7]. The third method works in an obvious way, but interestingly enough, is often ignored by researchers. In fact, many people believe that the efficiency of numerical optimization algorithm can be improved if human interactions in the design procedure are minimized. Arguably, product design optimization is different from pure mathematical optimization; in fact, product design is a decision making process, in which designers' experience should play an important role. On the whole, the basis of the design process is trial and error, and the success of the final design largely depends on the knowledge and intuition of the designer. However, most current optimization tools do not support the interaction between designers and the design problem. On one hand, it takes an optimization tool a long time to resolve a converged result; on the other hand, designers are not involved in the optimization process and cannot control the quality of the so-called optimal result. We believe a product design system should be developed to give designers the opportunity to guide the design and optimization process using known engineering rules and their experience. By using human interaction in the design procedure, high fidelity models like CFD will be able to display their ability to the fullest when coupled with numerical optimization methods. This statement is especially true when EAs are used as the optimization method because a healthy initial population can help reach the global optima much faster. Also, according to Parmee [8], appropriate integration designer interaction can result in the development of prototype evolutionary design tools that provide powerful extension to design team activity by supporting rapid, extensive exploration and stimulating innovative reasoning. The hypothesis was that initially ill-defined conceptual design problems can evolve through the close interaction of the designer with machine-based adaptive search process. The approach involves the capture of designer experience and intuition within evolutionary and adaptive search processes through an iterative designer/machine-based search within a relatively ill-defined design space [8]. The intention was to investigate the EAs' utilization as generators and gatherers of optimal design information.

3.2 Requirements and solutions for a virtual engineering CFD-based interactive design tool

Regarding the undergoing project of the virtual engineering design tool, the key to successfully implementing the new design tool is the need of rapid engineering analysis enabling interactive manipulation and interrogation of the engineering component or system. Therefore, not only the rendering, intuitive interaction, and geometry modification have to be very fast, but also the simulation time must be very short and a nearly on-line response of the simulation results is required. It is important that both the numerical optimization methods and fluid flow methods are accurate for shape optimization to work properly. Because the flow field inside a thermal system is usually highly three-dimensional and extremely complex, three-dimensional Navier-Stokes computations are essential, making design optimization based on three-dimensional Navier-Stokes computationally expensive. By introducing users into the design loop and implementing the new fast calculation algorithm [7,9] in the design system, this design tool will significantly reduce the time required by the traditional design method. Also, attention should be paid to the accuracy of numerical methods and verification of the models.

One important aspect, from the designer's perspective, is to know whether one idea out-performs another. Reliable approximations to the altered CFD flow can provide the qualitative results that determine whether the changes are effective. This requires that the CFD simulation is able to run interactively and to return results once complete. Typically the CFD analyst would make design changes requested by the engineer to the CFD model and once the simulation is complete would give the engineer two-dimensional plots of the results to examine. This cycle usually causes a disconnection between the CFD analyst, who produces the CFD models, and the engineer designing the system from the results of the CFD models. This disconnection typically occurs because there is no medium in which the CFD analyst and engineer can collaborate and move the design of the system forward. To break down this barrier between the analyst and the engineer, a solution is proposed that couples CFD software with a virtual engineering tool suite called VE-Suite. VE-Suite is a software package developed by Iowa State University. It is an open source library of tools that enables the engineering analysis and design process to take place in a virtual environment. It serves as the high-level support tool for engineers who want to transform their traditional applications into virtual engineering applications. The details of the software architecture and implementation of VE-Suite are discussed in Section IV.

A reliable and efficient CFD-based virtual engineering interactive design optimization tool has been developed using VE-Suite; the EA code is used for efficient and robust design optimization. To represent flow fields accurately and produce reliable designs, a three-dimensional Navier-Stokes is used for flow field analysis. This study demonstrates that the present method offers a promising approach to designers to design a better machine, while shortening design cycle and reducing design costs.

IV. Implementation

The components of the previous sections are to be combined within the virtual engineering system illustrated in Figure 3. It is difficult to effectively generalize the implementation process due to the great diversity in the manner in which the process is carried out and the resulting artifacts. The current discussion, though potentially applicable over a wide range, is intended to focus on building a virtual engineering interactive design tool. The discussion will focus on system pattern, the communication between different components, and the VE-Suite software package.

4.1 System pattern

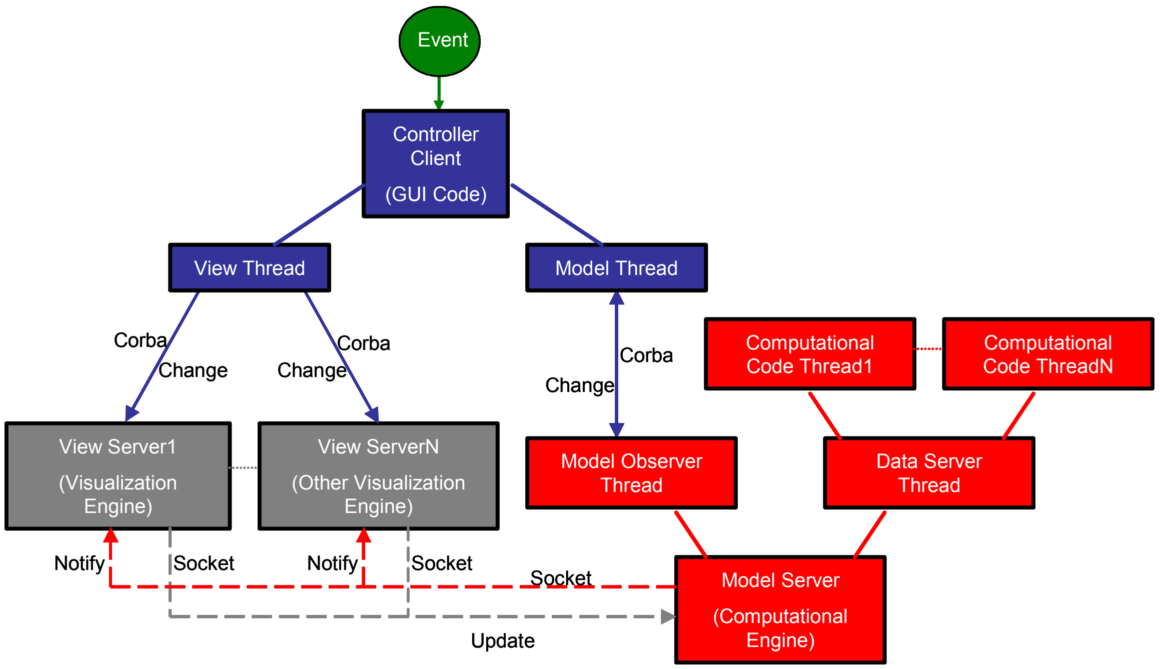


Figure 3 The Schematic View of the System

A design pattern that is very well suited to fulfill the requirements of the general virtual engineering application is that of a Model-View-Controller (MVC) pattern. Figure 3 shows the schematic view of the system. The MVC paradigm is a classic design pattern way of breaking an application into three parts: the model (red squares), the view (gray squares), and the controller (blue squares). The user input, the modeling of the external world, and the visual feedback to the user are separated and handled by controller, model, and view objects. The controller interprets mouse, keyboard, or other input events from the user and maps these user actions into commands that are sent to the model and /or view to effect the appropriate change. The model manages the behavior and data of the application domain and responds to requests for information about its state. A controller is the means by which the user interacts with the application. A controller accepts input from the user and instructs the model and view to perform actions based on that input. Whenever a controller changes a model's data or properties, all dependent views are automatically updated. Similarly, whenever a controller changes a view, the view gets data from the underlying model to refresh itself.

4.2 Communication and data transfer

In our system, the various components can run on different machines. The current architecture of this design allows the model object to run on a Linux cluster, which is cheaper and more flexible than an SGI machine, the view server on an SGI machine to utilize its graphical performance, and the controller client on a portable Tablet PC. Also, in order to be able to reuse the codes independently, the relationship among the model, view, and controller has to be decoupled. This strategy also allows the display engine to function regardless of the source of the data. Obviously, some means of coordination among the three objects is required to allow a complex environment of this kind to be used.

The virtual engineering software tool is very general in nature and the communication between the model and the control objects is based on a simple client/server (observer/subject) model. The client is the controller (user interface) process, which can connect to the model server to manipulate the underlying computational simulation. It accepts interaction commands from the user and enacts changes that affect the application's execution. For interactive design, the controller client also receives information from the model server and displays the information to the user via the view object. In its simplest form, where only one application has to be controlled, a single server process can be used to control and perform access to the model server. To integrate multiple applications in the environment, the client server model has to be extended. One possible solution is to create one separate thread of execution (model observer thread) of the application to supervise the requests from the client and one centralized data server for the interoperation and application to user interface communication and data transfer. The same model was used for the communication between controller client and the view servers. The communication between the three components is done via UNIX sockets and CORBA (Common Object Request Broker Architecture), and threading is done using the high-level threading API provided in VPR, the VR Juggler Portable Runtime Library.

4.3 VE-Suite

The backbone and integration platform of the virtual engineering design tool is VE-Suite, an open source software package that is under active development at the Virtual Reality Applications Center at Iowa State University. To enable performance on multiple operating systems and immersive technology platforms, VE-Suite is built upon VRJuggler, OpenGL Performer, and Kitware's Visualization Toolkit (VTK). VE-Suite is designed to be lightweight, portable, and efficient. It has the capability to handle not only CFD data, but also other engineering data such as FEA analysis results, experimental data, etc. All these features make it powerful while maintaining the ease-of-use by hiding most of the programming complexities from the user. Thus, designers can investigate the design problem from different disciplines and compare the experimental data with the analysis result. VE-Suite is introduced in detailed by McCorkle and Bryden in [10]. Currently, VE-Suite is capable of supporting the aforementioned modules.

4.4 Putting it all together

The current status of the virtual engineering interactive design tool is that the EAs, CFD calculation, and VE-Suite are at an advanced stage of development. They can be used as stand-alone techniques or integrated in the manner indicated in previous sections. Integrating defining interfaces that accommodate software and equipment vendors' proprietary products into the software developed in this system is at an early stage. This virtual engineering design tool is applied to a simplified pipe design case in the following section. Applying this tool to this simplified model will provide a proof of concept and an essential test bed; it will also enable developers to focus on the methodology and keep the effort tractable.

V. Case Study: Coal Pipe Design

In a coal-fired power plant, pulverized coal, using air as a transport medium, is pneumatically transported toward different burner nozzles by splitting the large pipe into small pipes through bifurcators or trifurcators. The combustion efficiency of the burner is dependent on matching the air and coal in these pipes. The increasingly tight emission control also makes the balance of air and coal a very critical factor for the success of the power plant.

If this gas-solid mixture flows along a straight pipe, the solid phase may separate from the air because of gravity. If the mixture flows around a bend, the inertia of the solids will also force them towards the outer wall of the bend, and this volume with high particle density tends to stay together. This phenomenon is termed as "coal roping". In a power plant, the paths of the individual coal pipes are complex with many bends, so uneven coal distribution is inevitable. There are several options for balancing the coal flow from the pulverizer to individual burners. Devices known as riffle distributor control the coal distribution of the pipe system. Schneider [11] reported that without coal rope in front of a distributor, the particle distribution at the inlet cross-section of a distributor is uniform and the mass flow is divided relatively uniformly into both outlets. However, if coal roping occurs in front of the distributor, particle distribution is not uniform over the pipe cross-section and the coal will no longer be evenly divided between the two outlets. As a result, the fuel inputs to the burner are not balanced, combustion efficiency is greatly decreased, and emissions increase. This problem must be handled when designing the pipe system.

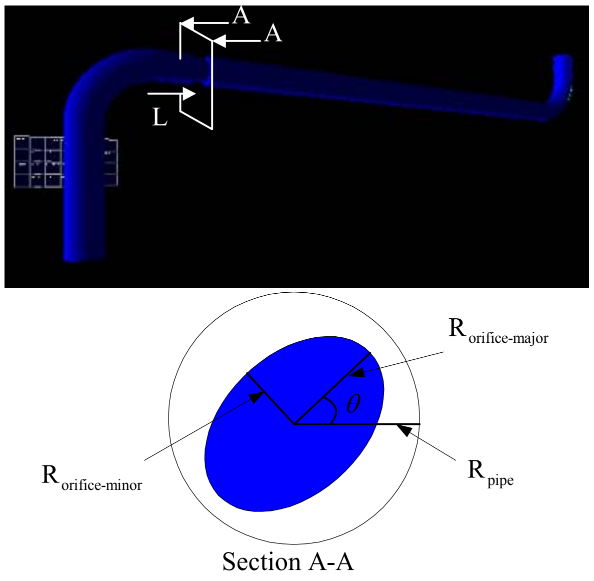


Figure 5 Sketch of Testing Pipe Geometry

Table 1 Geometry Sizes of the Testing Facility

Pipe Diameter, d (m)	First Vertical Pipe Length	Horizontal Pipe Length	Second Vertical Pipe Length	Elbow Radius
d =0.154	**5 × *d***	15 × d	3 × d	3 × d

Shown at the upper part of Figure 5, the basic pipe geometry consists of a vertical pipe with a length of five pipe diameters, the elbow section, a horizontal pipe with a length of 15 pipe diameters followed by another elbow, and a vertical pipe with a length of three pipe diameters. Both elbows are 90˚ and the radius of the elbows is three times the pipe diameter. Table 1 shows the geometry dimensions of this facility. In order to break the coal rope, one elliptical orifice is installed somewhere in the long horizontal pipe. It has been known that in single-phase flow, fluid that enters through a noncircular inlet entrains more fluid than does comparable fluid entering through a circular inlet. If this is also the case for gas-solid flow, the stronger mixing mechanism may be able to increase the dispersion rate of the coal rope. Although there is no literature that discusses using elliptical orifices in the coal pipe, in this case, the cross-section can be elliptical. The lower part of Figure 5 shows the parameters determining the shape of the orifice. The design parameters in this case include the location of the orifice, L, the orientation of the orifice, θ, and the ratio of the radius of orifice ($R_{\text{orific-major}}$) and the pipe, $\tilde{r}$. As the first step to investigate the gas-solid two-phase flow problem in this paper, we only analyze the continuous phase of gas flow. The objective of the pipe design optimization is to minimize the gas velocity difference at the exit surface of the pipe. The results of the single-phase flow can be used in future work, when Lagrangian particle tracking is used to analyze the gas-solid two-phase flow.

The optimization problem is then formulated as:

Minimize: (1)

$$D = \frac{K\sum_{k=1}^{K}\sqrt{\left(v_k - \frac{\sum_{k=1}^{K} v_k}{K}\right)^2}}{\sum_{k=1}^{K} v_k}$$

where: $$0^o < \theta < 90^o \quad (2)$$
$$2 \cdot d < L < 14 \cdot d \quad (3)$$
$$0.7 < \tilde{r} < 0.9 \quad (4)$$
$$\frac{S_{orifice}}{S_{pipe}} = 0.8 \quad (5)$$

In the optimization process, the virtual environment design tool should be able to generate the grid model, supply the model to the CFD solver, and then post-process to evaluate the fitness of the design. The CFD analysis is carried out using the commercial package Fluent^{TM} and its preprocessor GAMBIT™. The basic implementation process is as follows:

Grid Generation — A GAMBIT journal file is used to automate grid generation; GAMBIT can be used to generate a journal file, but only for the fixed parts of the model. Thus a C++ program is written for the automatic generation of three-dimensional grids for the changing part of the models; in this case, the changing parts are the size, shape and location of the orifice. The journal file takes the number of grid points and the clustering parameters as its input. The wall and boundary conditions are also specified in the journal files and stored in mesh files. The C++ program combined its output with the GAMBIT generated journal file to create a single journal file for each case. Therefore, the complete process of grid generation can be invoked in a batch mode of operation using a single GAMBIT journal file.

Analysis Using Fluent™ — The process of grid generation is integrated with the solver through the FLUENT journal files. The mesh files contain the grid and boundary conditions. The Fluent journal files specify input file format, flow model specifications, and parameters such as the under relaxation factor, boundary conditions, simulation details, and the output format. The renormalization group RNG $k-\varepsilon$ turbulence model is used to simulate turbulent gas-particle flows through the pipe. The main reason for choosing the RNG $k-\varepsilon$ model is due to its performance over the standard $k-\varepsilon$ model at predicting the streamline and radial velocity components and Reynolds shear stresses within a 90˚ circular pipe bend [11].

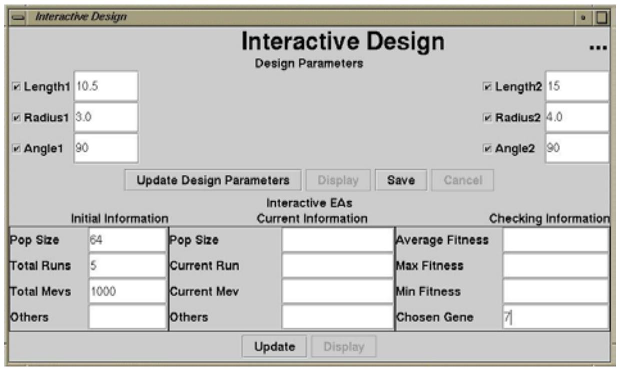
Fitness Evaluation — The user-defined functions (UDF) provided by FLUENT™ are used to calculate the fitness value at the end of each CFD iteration. A program is developed using the FLUENT API to access the flow variables, calculate the fitness, and store the result in an output file, which may be accessed by the optimizer. The fitness values are used to stop the CFD iterations when the convergence criterion is satisfied.

Post-Processing at the End of Inner-Cycle — Since VE-Suite uses VTK format in its graphic engine[10], post-processing in this case is transforming the CFD analysis result into VTK files for visualization.

EA Optimizer — The standard EA optimizer includes several components: selector, crossover, mutation, and replacement. At first, two creatures are chosen from the population using the roulette wheel method. After that, random point crossover and Gaussian mutation are used to generate two children. Then, two random creatures from the last generation, except the two creatures with the highest fitness values, are replaced with the newly generated creatures. More information about EA can be found in [12]. For this paper, we carried out three runs with population size set to 32. Each run had a different initial population and stopped after 770 generations. In the first two runs, the initial population was selected arbitrarily from the design space. In the third run, designers defined the first 24 population based on their experience gained from the interactive process; the computer randomly generated the remaining eight populations. With all three runs, compute randomly generated the crossover operator probability p_c and the mutation operator probability p_m.

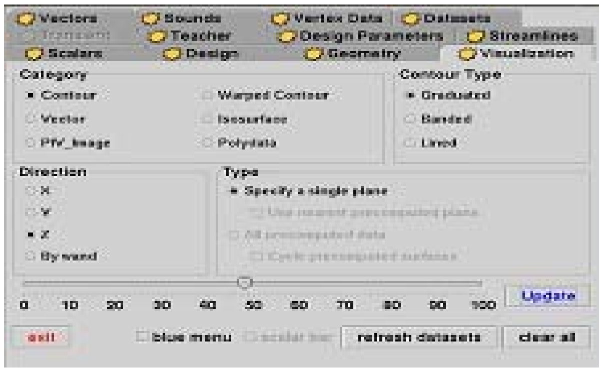
Connection to the EAs optimizer — A simple script file is coded to call the above activities sequentially. Because the prototype is designed for short-term demonstrations, we made it simple. However, the graphical user interface is flexible. In the first step of choosing an initial population, the EA optimizer is not called. The user has the ability to change the design parameters through the window shown in Figure 6. When the "Update Design

Parameters" button is clicked, the user-defined parameter values are sent to activate the computational engine, which can change the geometry of pipe according to the desired design parameters and carry out the CFD calculation automatically. Once the simulation is done, the results are visualized using a wide variety of visualization tools (Figure 7). Users can decide whether this is a healthy design candidate or not. After this step, the system connects the EA optimizer and conducts multiple CFD analyses. The design parameters are changed automatically, while the visualization function is no longer called. During this optimization process, users can also keep tracing the searching process by clicking the "Update" button.



✓ Length1	10.5			✓ Length2	15
✓ Radius1	3.0			✓ Radius2	4.0
✓ Angle1	90			✓ Angle2	90
Pop Size	64	Pop Size		Average Fitness	
Total Runs	5	Current Run		Max Fitness	
Total Mevs	1000	Current Mev		Min Fitness	
Others		Others		Chosen Gene	7

Figure 6 VE-Suite Design Interface



Category: [✓ Contour]
Contour Type: [✓ Graduated]
Direction: [✓ Z]
Type: [✓ Specify a single plane]

Figure 7 VE-Suite Visualization Interface

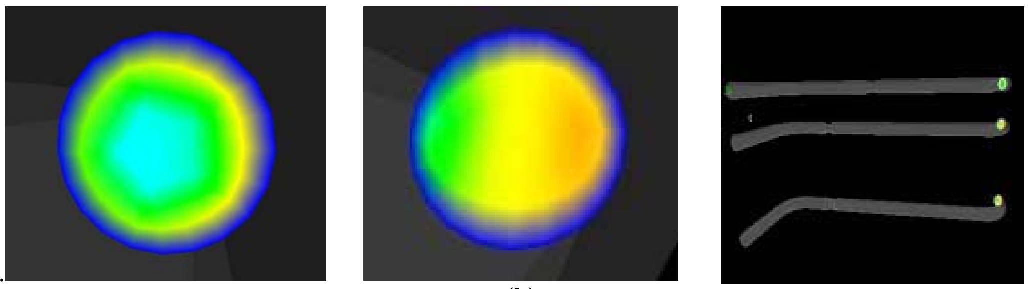

(a) (b) (c)

Figure 8 Velocity Distribution Contours

The contours of the velocity distribution at the exit in a baseline pipe and the EA-searched best result are shown in Figure 8 (a) and (b), respectively. Figure 8 (c) shows that several design results can be loaded into the virtual environment for comparison. The variations of the best individual's fitness during the evolution process in the run are shown in Figure 9. It shows that with a good initial population, EA can find the optimal solution much faster. In our case, the required time is reduced by 50%. Table 2 shows that the time reduction by using the virtual engineering design tool can be as high as 97%.

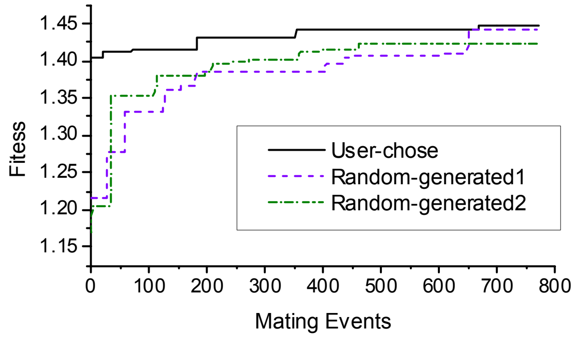


Figure 9 Fitness Variations in the Evolution Processes

Table 2 Comparison between the Traditional Designs and the Virtual Engineering Design

		Initial pop size	Mating events	Total calls for inner cycle	Time required to complete one inner cycle (min)	Total time (h)	Time reduction
Traditional numerical design	Original two-cycle	32	650	2632	15	658	
Virtual engineering design system	Modified two-cycle	32	650	2632	0.8	35	95%
	User defined initial population	58	330	1378	0.8	18	48%
							97%

VI. Conclusion

This paper has discussed the basic requirements and applications of a general virtual engineering application. Based on these requirements, a basic software structure (MVC pattern) and realization of a virtual engineering interactive design tool was provided. The virtual engineering interactive design tool facilitates multidisciplinary collaborative design in an immersive interactive real-time environment. VE-Suite is a suite of virtual engineering tools; it integrates existing design and analysis tools into a common interface that allows for the modification and visualization of design cases in real time. Using VE-Suite and other virtual engineering technologies, the design process of a coal-pipe for coal transport was greatly enhanced by shortening the required design process time and allowing the designer a "hands-on" approach to tweaking the design parameters. This enhanced both the designer's personal knowledge and intuition and the overall result of the design process by allowing the designer more time to explore "what if" cases, among other things. In the future, development of VE-Suite will continue in the direction of facilitating more tools from a wide variety of scientific backgrounds into a common interface so that virtual engineering can accommodate more disciplines in its multidisciplinary collaborative approach.

References

[1]Xiao, A. and Bryden, K. M., "Virtual Engineering: A vision of the next-generation Product Realization Using Virtual Reality Technologies," *Proceedings of ASME Design Engineering Technical, Computers in Engineering Conference*, DETC2004/CIE-57698.

[2]Diachin, D., Freitag, L., Heath, D., etc., "Collaborative Virtual Environments Used in the Design of Pollution Control Systems," *International Journal of Supercomputer Applications and High Performance Computing*, Vol. 10, 1996, pp. 223-235.

[3]Ryken, M.J. and Vance, J.M., "Applying Virtual Reality Techniques to the Interactive Stress Analysis of a Tractor Lift Arm," *Finite Elements in Analysis and Design*, Vol. 35, 2000, pp.141-155.

[4]Kihonge, J.N., Vance ,J.M., and Larochelle, P.M., "Spatial Mechanism Design in Virtual Reality with Networking," *Journal of Mechanical Design*, Vol. 124, 2002, pp.435-440.

[5]Perles, B. and Vance, J.M., "Interactive Virtual Tools for Manipulating NURBS Surfaces in a Virtual Environment," *Journal of Mechanical Design*, Vol. 124, 2002, pp.158-163.

[6]Rossi, L.F. and Sohos, G., "Interactive Simulation of Contaminant Evolution through Porous Media," *Future Generation Computer Systems*, Vol. 15, 1999, pp.477-484.

[7]Huang, G.H., Xiao, A., and Bryden, K.M., "A Virtual Engineering Tool for Product Design Using High Fidelity Analysis Models," Proceeding of ASME International Mechanical Engineering Congress, IMECE2004-61832.

[8]Parmee, I., Cvetkovic, D., Bonham, C., and Packham, I., " Introducing Prototype Interactive Evolutionary Systems for Ill-defined, Multi-objective Design Environments," *Advances in Engineering Software*, Vol. 32, 2001, pp.429-441.

[9]McCorkle, D.S., Bryden, K.M., and Carmichael, C.G., "A New Methodology for Evolutionary optimization of Energy Systems," *Computer Methods in Applied Mechanics and Engineering*, 2004, (to be published).

[10]McCorkle, D.S., Bryden, K.M., and Kirstukas, S.J., "Building a Foundation for Power Plant Virtual Engineering," *Proceedings of the 28th International Technical Conference on Coal Utilization and Fuel Systems*, 2003, pp.118-127.

[11]Schneider, H., Frank, T., Pachler, D.K., and Bernert, K., "A Numerical Study of the Gas-Particle Flow in Pipework and Flow Splitting Devices of Coal-Fired Power Plant," *10th Workshop on Two-Phase Flow Predictions*, 2002, pp.227-236.

[12]Goldberg, D.E., "*Genetic Algorithm in Search, Optimization and Machine Learning*," Addison-Wesley, MD, 1989.