A DATA-CENTERED FRAMEWORK FOR AN ASSISTANT-BASED SCIENTIFIC VISUALIZATION SYSTEM[1]

Kristina D. Miceli

NAS Technical Report 94-006, December 1994

Numerical Aerodynamic Simulation (NAS) Systems Division
NASA Ames Research Center
MS T27A-1
Moffett Field, CA 94035-1000
kmiceli@nas.nasa.gov

Abstract

This report presents the development of an assistant-based visualization system (or "visualization assistant"), based on a data-centered framework, that supports the production of effective graphics for scientific visualization. The framework is composed of a data model, a user model and a machine model that represent the data, the scientific user and the software/hardware environment. Each model contains knowledge related to the production of effective visualizations, including rules from perception, graphic design and the scientific domains. The goal of this research is twofold: 1) to present scientists with a productive visualization tool that allows them to concentrate on the task at hand, and 2) to generate effective visual representations that aid in interpretation. Both of these goals are accomplished by taking into account the data analysis task, the type of data to be displayed, the capabilities and preferences of the scientist, software/hardware characteristics, and visualization knowledge. This report describes the framework for the visualization assistant, the design process, the prototype system, and areas for future research.

[1]This report has also been published as a doctoral dissertation through the University of Colorado, Boulder, Department of Computer Science, December 1994.

Mickus-Miceli, Kristina Diana (Ph.D, Computer Science)
A Data-Centered Framework for an Assistant-Based Scientific Visualization System
Thesis directed by Adjunct Research Assistant Professor Gitta Domik

Scientific visualization is an enabling tool that assists scientists with the visual analysis of large datasets such as those produced by computer simulations. Many visualization systems are currently available to help scientists with their data analysis tasks, each providing different functionality for scientists to use when analyzing their data. However, along with functionality comes complexity and many scientists are unwilling to spend much time learning to use visualization systems. As a result, scientists either use visualization systems in limited ways or in a trial-and-error manner, neither of which is very productive. To compound this problem, scientists unfamiliar with visualization concepts may create representations that are useless or misleading.

This research presents the development of an assistant-based visualization system (or "visualization assistant"), based on a data-centered framework, that supports the rapid production of effective graphics for scientific visualization. The framework is composed of a data model, a user model and a machine model that represent the data, the scientific user and the software/hardware environment. Each model contains knowledge related to the production of effective visualizations, including rules from perception, graphic design and the scientific domains. The goal of this research is twofold: 1) to present scientists with a productive visualization tool that allows them to concentrate on the task at hand, and 2) to generate effective visual representations that aid in interpretation. Both of these goals are accomplished by taking into account the data analysis task, the type of data to be displayed, the capabilities and preferences of the scientist, software/hardware characteristics, and visualization knowledge.

To demonstrate the feasibility of this approach, a prototype visualization assistant called MDV, or MultiDisciplinary Visualizer, was developed at the Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center. The application domain was computational aerosciences, involving multidisciplinary data from computational fluid dynamics and structural dynamics. The system was designed with the close involvement of scientists performing the simulations. This dissertation presents the framework for the visualization assistant, the design process, the prototype system, and areas for future research.

ACKNOWLEDGEMENTS

I am very thankful to many people who provided me with their time, energy and insight. I would especially like to acknowledge:

- My principal advisors, Gitta Domik and Sam Uselton, for their guidance, support and encouragement.
- The remaining members of my committee, Clayton Lewis, Wayne Citrin, Rick Hull, Ben Zorn, Liz Jessup, and the CU Computer Science Department, for being patient and understanding regarding my off-campus status.
- SuperGlue Developers, Jeff Hultquist and Eric Raible, without whose work this project would have no foundation.
- Multidisciplinary simulation researchers, Eddy Pramono and Sisira Weeratunga, for being enthusiastic and patient users.
- The Applied Research Branch of the Numerical Aerodynamic Simulation Systems Division at NASA Ames, for providing a stimulating and creative work environment.
- Computer Sciences Corporation, for sponsoring my graduate fellowship position as well as providing me with funding for travel.
- Additional people I've met through the course of my work, especially John Dannenhoffer, who have provided real world insight.
- To my extended family, for making me feel like I was smart enough to pull this whole thing off.
- Finally, to my husband, for understanding what it takes to reach your goals.

CONTENTS

FIGURES

CHAPTER 1
INTRODUCTION

Scientific visualization is a useful, but complex, tool for scientific data analysis. Due to the complexity in understanding and using scientific visualization, the benefits of scientific visualization are not completely realized. As a result, data analysis productivity is not maximized.

This thesis describes the development of an assistant-based scientific visualization system, or a "visualization assistant". A visualization assistant guides the scientist with the process of designing and generating effective visualizations of his/her data. The data analysis process can be more productive with the help of a visualization assistant.

Productivity is enhanced in two ways. First, generating visualizations is less time-consuming because the scientist is not required to learn complicated visualization systems. As a result, the scientist can concentrate his/her attention on the task at hand and thus, be more productive. Second, more effective visualizations are designed by taking into account visualization knowledge from perception, graphic design, and the scientific domain. Designing a more effective visualization may aid in interpretation and lead to a faster conclusion.

The research presented in this thesis is a contribution to the design and development of visualization systems. The contribution comes in two parts: a high-level framework and a user-tested implementation. In the design of the visualization assistant, the development of a high-level, structured framework is required to define the necessary information sources that must comprise such a system. The development of this framework provides a structure from which systems can be built and expanded. This structured approach to visualization, as opposed to current *ad hoc* approaches, is a step towards understanding the visualization process as a whole.

A user-tested prototype system has been built based on this framework. Feedback from scientists has been obtained regarding the visualization assistant approach and its place in the development of next-generation visualization systems. Implementation considerations have been evaluated and these results are presented so that future systems can take advantage of the information gained from this experience.

The framework presented in this thesis consists of a data model, a user model, a machine model and a knowledge base. This framework is original in that it is the first to combine several of the key information sources necessary for the design of effective visualizations. This thesis asserts that the above specified information sources are necessary, together with the data analysis goal of the scientist, to create an effective visualization assistant. In addition, scientists were involved throughout the design process, leading to a prototype system designed based on user feedback. Previous research did not incorporate a great deal of participation from end users.

Concepts from the database and artificial intelligence (AI) fields were used in designing the visualization assistant. The technology from these fields is very promising and can be applied to many different applications. The database and AI concepts used in this thesis were helpful in modeling and implementing the visualization assistant. However, these technologies were only used to support the design of the visualization assistant. The use of these technologies is not intended to present significant contributions to either the database or AI fields.

Scientific visualization and its role in the data analysis process is presented in the next section. The difficulties faced by scientists using current systems are described and possibilities for next-generation visualization systems are investigated. The following section provides a high-level introduction to the thesis project. The final section on related work identifies key issues and shortfalls of research in this area.

1.1 Scientific Visualization

Computer simulations of physical phenomena, such as fluid flow over a structurally deforming aircraft wing, are producing increasingly larger and more complex datasets. To obtain insight into the phenomena modeled in these datasets simply by studying the numbers is difficult, if not impossible.

The purpose of computing is insight, not numbers. *Richard Hamming[31].*

The technology to graphically display scientific data is essential to provide this insight. This technology, termed scientific visualization [45], takes advantage of the powers of the human visual system in interpreting scientific data. The human visual system possesses two characteristics that make scientific visualization such an effective tool. First, the large bandwidth of the human visual system aids in the rapid browsing of large amounts of scientific data. Second, the visual system is proficient at detecting pattern and structure. This skill can be applied to scientific data. Because of these characteristics, scientific visualization can assist the scientist with the analysis of the large and complex datasets currently being produced.

Scientific visualization is an important tool in the scientist's workbench. It can serve many roles, including:

- A debugging tool for developing simulation software,
- An analysis tool for examining, correlating and comparing datasets,

- A validation and verification tool for checking data correctness,
- A communication tool for conveying concepts and phenomena between scientists, and
- A presentation tool for presenting work to the research and funding communities.

1.1.1 Current Visualization Systems

The utility of scientific visualization has been demonstrated by its general acceptance by the scientific community. This acceptance is evident from the many visualization systems that exist to analyze data from applications such as fluid dynamics and medicine [4, 42, 63, 68]. Each system possesses the functionality to explore data from one or more of the above mentioned applications. However, along with greater functionality comes increased complexity and many scientists are unwilling to spend much time learning to use visualization systems. As a result, scientists either use visualization systems in limited ways or in a trial and error manner, neither of which is very productive. The complexity of current visualization systems has caused many scientists to avoid using technology that could potentially be very valuable to them.

Not only is time required in learning to use visualization systems, but knowledge about visualization technology is beneficial. If a scientist does not have a background in visualization, he/she could design images that lead to confusion or misinterpretation. This background includes knowledge about the different types of visualization mappings available and which mappings fit which types of data. A visualization mapping, or visualization technique, is a method for transforming data into a visual representation of the data[2]. A visualization is the final image produced by a mapping.

A scientist often seeks assistance from a visualization expert to help select and apply a visualization mapping. If such an expert is not available, the scientist must design and generate his/her own visualizations. The process of designing and generating a visualization of data from a computational fluid dynamics (CFD) simulation is described in the following sample scenario.

Data Analysis Scenario

A fluid dynamicist is interested in exploring a velocity vector field computed over the wing of an airplane. The fluid dynamicist must:

[2]Examples of visualization mappings include contour lines, color contoured (or colormapped) surfaces, and vector arrow plots.

1. Determine what visualization mappings are useful in viewing velocity fields,
2. Find and learn to use the software that implements such mappings,
3. Correctly select from the many options to enhance the image such as color, lighting, viewpoints, etc.

A more concrete scenario describes the steps necessary to perform this task using the fluid dynamics visualization system FAST (Flow Analysis Software Toolkit[4]).

Data Analysis Scenario using FAST

1. Start Up: Run the FAST program.
2. Read Data: Read the desired datasets into FAST using the *File IO* module. This step involves a minimum of five button presses (with additional button presses if the file format is more complex), performed in one window.
3. Calculate Data: Open the *Calculator* module and compute the velocity field from the existing data fields. This step involves opening another window, and performing a minimum of one button press.
4. Designing the Visualization: Designing the visualization first involves selecting the appropriate visualization mapping. Selecting the appropriate mapping requires general visualization knowledge, knowledge about the FAST system and its available mappings, knowledge about standard visualization mappings used in the physical domain and knowledge about the characteristics of the data. Particle traces[3] are selected because they are useful in depicting fluid dynamics velocity vector fields.
5. Generate the visualization: The simplest implementation involves generating a wireframe mesh of the geometry (using the *Surfer* module) and creating several particle traces (using the *Tracer* module). Selecting seed points[4] for the particles is somewhat of a trial and error process, requiring the user to adjust sliders and viewpoints to understand the relationships between the geometry and the particles. Performing this visualization mapping requires opening two windows, five button presses and the adjustment of five sliders. The number of button presses and slider adjustments can increase greatly due to the trial and error process of placing seed points.

[3]A particle trace (also called a streamline in a time-independent flow field) is the path a particle travels in the computed flow (velocity) field. Particle traces provide information about the directionality of the flow and structure in the flow field.

[4]A seed point is the originating position for the particle trace.

6. Adjust visual attributes: Completing the design of the visualization requires adjusting visual attributes such as lighting, rendering style (*i.e.* lines versus polygons) and color. Rendering attributes must be adjusted for each component of the visualization, namely, the wireframe mesh and the particle traces. This step is often optional, since default visual attributes are typically sufficient.

The resulting visualization is shown in figure 1.1. The complexity of the visualization environment is depicted by the many panels, buttons and sliders the scientist must use in generating a visualization. However, the FAST system is very functional and provides the scientist with many different tools to use for data analysis.

The process of generating visualizations can be complicated and time consuming, as shown in the previous example. The rapid generation of visualizations is important due to the exploratory nature of scientific data analysis, where visualizations are often viewed briefly and then discarded. If too much time is required to generate a visualization, including both the time to specify the visualization mapping and the time to compute and render the visualization, scientists may abandon the system. Scientists need a system to support the exploratory process, allowing them to generate images easily and rapidly so they may focus on their scientific problem.

1.1.2 Next Generation Visualization Systems

Research in human/computer interaction (HCI), coupled with representational techniques from artificial intelligence and database theory, has shown promise in helping to reduce the complexity of sophisticated software systems. The goal of research in HCI is to develop user interfaces that hide software complexity, providing a tool that allows the user to specify his/her task in a more intuitive manner.

Assistant-based systems [25], intelligent agents [18], and intelligent user interfaces [64] are related user interface concepts developed to guide the user through complicated environments. The user interface, enhanced with knowledge about the given application domain, provides the user with advice and assistance. As a result, users can perform their tasks in a more productive and effective manner. The user never forfeits control of the system. Rather, the user benefits from the knowledge contained in the system, accepting and using the information or choosing to ignore it. The cooperative problem solving approach places emphasis on using knowledge-based representational techniques to augment and empower human users, rather than replace them.

The above-mentioned user interface approaches can help manage the complexities involved in scientific visualization. The ability to express data

Figure 1.1: The FAST visualization system.

queries more intuitively can make visualization tools easier to use for the average scientific user. For example, the visualization in the previous section might be generated with the query: "Explore the velocity field over the wing in dataset XYZ." This query is answered by the MDV (MultiDisciplinary Visualizer) visualization assistant described in this thesis. The scientist inputs the query via a menu-based interface and MDV selects an appropriate visualization and renders it to the screen. The scientist has the ability to interact with the visualization, changing viewing parameters and the position of the streamlines with the mouse. If interested, the scientist can click on the visualization and a popup menu will appear which describes the attributes of that visualization. The scientist can modify these attributes as desired. The resulting image is shown in figure 1.2.

Several benefits arise when providing assistance to scientists with their visualization task. First, scientists are often unaware of the constraints imposed by computer software, hardware and the human visual system when designing a visualization. If this information is incorporated into the assistant-based system, the resulting visualization can be better suited for the task at hand. Second, the amount of time spent learning how to use visualization software and generating visualizations is reduced, thus giving scientists more time to dedicate to their research.

The need for such a system is expressed in the recent report by the Office of Naval Research Advisory Panel on Scientific Visualization [56]. The report, entitled "Research Issues in Scientific Visualization" cites the need for user interfaces to scientific visualization systems that make visualization tools more productive by incorporating information about the data, the user, system characteristics and the visualization task. The report mentions the development of "visualization assistants" that can help the user to quickly design and generate meaningful visual representations.

1.2 Research Overview

Based on the background and motivation described in the previous sections, research in developing a visualization assistant was performed in this thesis. An overview of this research, describing its basic premise and approach is presented in the following sections.

1.2.1 Thesis Statement

For effective and productive data analysis, a visualization assistant must contain information about:

- *the data to be visualized,*

Figure 1.2: The MDV Visualization System.

- *the user and his/her data analysis task,*
- *the hardware and software environment,*
- *the application domain,*
- *visual perception and graphic design.*

Determining what constitutes an "effective" or "useful" visualization is a difficult task. This has been approached by researchers by studying perception and graphical presentation. For example, Mackinlay [43] defined the words "effective" and "expressive" in attempting to describe a useful visualization. An "expressive" visualization fully encodes the data attributes by presenting all the relevant (and *only* the relevant) information to the user. The complementary word "effective" suggests that the technique exploits the capabilities of the output medium and human visual system, leading to a correct and quick judgement of the data.

This thesis asserts that several information sources are required to develop effective and expressive visualizations. The information sources identified by this research project include the data, the user, the hardware and software environment and expertise from the individual scientific domains, perception, and graphic design. Other sources of information may be necessary, but were not identified in this thesis. A data model organizes the data into an appropriate form so that information about the data and its characteristics can be accessed easily. A user model describes the goals and characteristics of the user so the system can better adapt to his/her requirements, abilities and preferences. A machine model is necessary to determine how each aspect of the computer, both software and hardware, impacts the resulting visualization and interaction techniques. A knowledge base contains principles from perception, graphic design, and expertise in the scientific domains. Information in the knowledge base can be coupled with information in the data, user and machine models, when determining an effective and expressive visualization.

This thesis also asserts that the data analysis task of the scientist plays an important role in the visualization assistant. Different tasks performed on the same data, by the same user and on the same hardware system can require different visualizations. For example, if the user (a scientist) is interested in accomplishing a task such as debugging a simulation, the visualization presented to him/her would be different from the scientist who was interested in generating a visualization for publication. MDV considers the data analysis task in the design of a visualization.

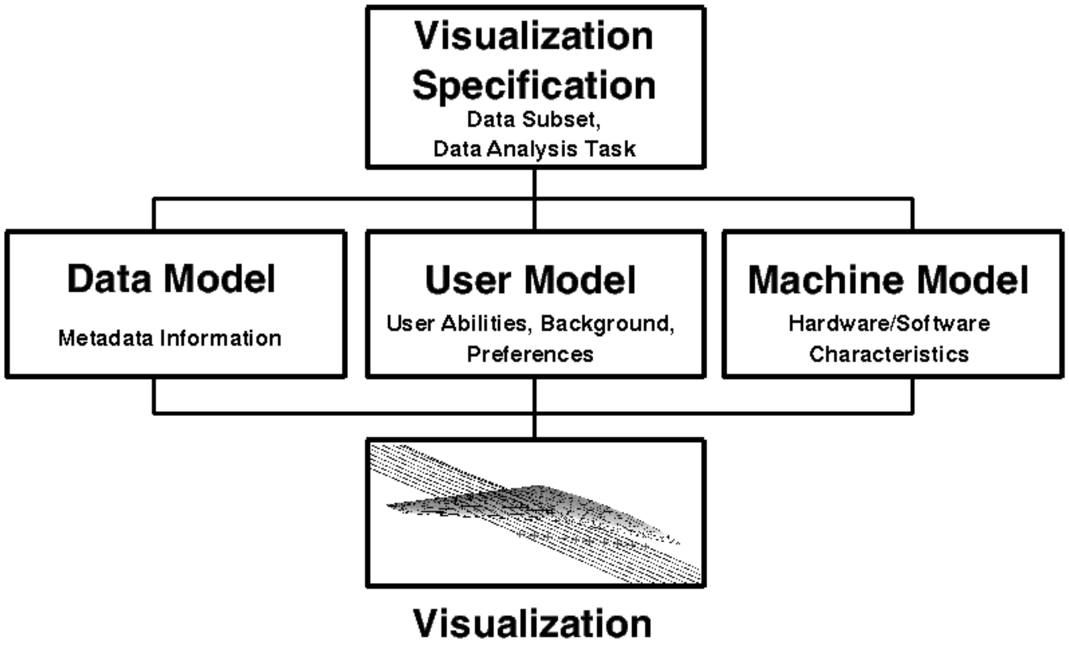

Visualization Specification
Data Subset,
Data Analysis Task
Data Model
Metadata Information
User Model
User Abilities, Background, Preferences
Machine Model
Hardware/Software Characteristics
Visualization

Figure 1.3: The Visualization Framework.

1.2.2 Approach

A visualization assistant was developed to help scientists with the rapid generation of effective and expressive visualizations. The visualization assistant is based on a framework that consists of a data model, a user model, and a machine model (shown in figure 1.3). The framework also contains a knowledge base of visualization information. The knowledge base is not a single entity *per se*. Instead, knowledge is incorporated into the relevant information model. Information in the framework is used by the system in designing a visualization based on a visualization specification (or query) presented by the scientist.

The three models were selected as the primary sources of information necessary for designing effective visualizations. The knowledge base provided additional information from perception, graphic design and the scientific domains. The data model represents the data contained in the scientific domains. The data for each domain contains the geometry of the simulated objects, the data variables that have been calculated in the simulation, and the attributes and relationships of the data variables.

To complement the data model, the user model describes the domain scientist and his/her characteristics. These characteristics include the scientist's background, preferences, and physiological capabilities. Background information about the scientist includes information such as his/her scientific domain and computer and visualization expertise. Preferences include a favorite color table or a preference for a particular style of interaction (*i.e.* textual versus menu-based). Physiological capabilities include factors such as a color blindness that might affect interpretation of color-coded images, or difficulty with fine-motor coordination that might affect using a mouse for direct manipulation.

The machine model incorporates information about the resources available in the scientist's computing environment. This information includes the characteristics and limitations of the software and hardware available for generating visualizations. For example, if the output is to be generated on a color printer, various color choices might be avoided because they do not reproduce well.

The knowledge base contains principles from perception, graphic design, and the individual scientific domains. This knowledge is combined with the information in the data, user and machine models, providing assumptions that guide with the selection of visualization mappings and their visual attributes. For example, if the scientist is interested in viewing a fluid dynamics velocity field, an assumption contained in the data model suggests using a particle trace because velocity is a vector field. If the scientist currently performing the data analysis is red-green colorblind, an assumption contained in the user model suggests using an appropriate colortable. If the output medium selected is a black and white printer, an assumption contained in the machine model suggests using a black contour lines and a white background.

Visualization knowledge is stored in the individual models and is accessed by the system based on the data analysis task of the scientist. The task is represented by a "visualization specification", which contains two components. The first component specifies the data the scientist wishes to view. This component includes selecting subsets of the data and field variables of interest. The second component identifies what data analysis task the scientist is interested in performing. MDV represents the task component by providing high-level task variables such as "debugging" and "presentation".

This framework defines the components that are critical for a visualization assistant. However, each component in the framework is an individual research project. As a result, a simplified version of the overall framework is studied in this thesis. The simplified version allows a thorough study of the principal component of the framework, the data model, and how the mapping of data to visualizations is performed using information in the knowledge base. The data model was selected as the main and driving component of the visualization framework because the characteristics of the data define what visualization mapping should be applied to the data. The information in the user and machine models is important in the selection of this mapping. However, the user and machine models only support and refine the mappings from data to visualization. Their impact can be studied as a later research project in the development of this visualization framework.

1.2.3 Application Domain

The visualization assistant was developed for visualizing computational aerosciences (CAS) data generated at the NASA Ames Research Center. Research in computational aerosciences involves the coupled simulation of the several physical disciplines that affect flight characteristics of next generation aeronautic vehicles, such as the High Speed Civil Transport (HSCT). These multidisciplinary simulations model disciplines such as fluid dynamics, structural dynamics, thermodynamics, and propulsion.

Many challenges are posed by multidisciplinary simulations in computational aerosciences. As the simulation technology evolves, supporting tools must also evolve to assist the simulation scientists. One of these tools is visualization, which assists scientists in debugging, exploring, analyzing and presenting simulation results.

Multidisciplinary simulations produce complex, heterogeneous datasets. The data is stored in different data formats, with different grid qualities and different physical field variables (*i.e.* velocity, pressure, deformation). To create a productive environment in which supporting tools do not hinder the scientist, implementation details should be hidden from the scientist. Data management and user interface studies are critical components to address these challenges and are addressed in this thesis.

For the prototype implementation described in this thesis, the data model characterizes datasets in multidisciplinary (fluid dynamics and structural dynamics) simulations in progress at NASA Ames [50]. The simplified user model characterizes the computational scientists involved in the project, their preferences and working environment. The simplified hardware model describes the scientists' working environment at NASA Ames. The knowledge base contains information about visualization mappings, perceptual and graphic design considerations and knowledge from fluid dynamics and structural dynamics domains.

The end product of this work is a prototype visualization assistant. The scientist makes a query to the system selecting data objects, physical field variables of interest, the desired output medium and his/her visualization task. The system processes this information and presents a visualization to the scientist, rendered and annotated. The scientist then explores the data via direct manipulation. "Popup menus", that are custom built to suit the attributes of the chosen visualization mapping, provide the scientist with additional information and functionality. For example, the scientist may want to know the properties of the data or why a particular visualization mapping was selected. The scientist may also modify some attributes of a visualization, such as color choices, using the popup menus. Finally, the scientist may return to a more traditional user interface at any time if they are not satisfied with the information provided by the assistant.

1.2.4 Design Philosophy

The design philosophy for this project was that of participatory, iterative design. Past experiences have shown that software projects are more successful when user input is solicited throughout the development cycle [9, 47].

Participation from the scientist provides valuable input to the direction of the project and influences the design of the user interface. Scientists' input was continuous throughout the project lifecycle through mechanisms such as interviews, informal meetings and demonstrations. Interaction with the scientist was also emphasized to obtain domain knowledge for the knowledge base.

The iterative process allowed for a continuous, step-by-step improvement over the project development cycle. The scientist had opportunities to suggest improvements on a usable prototype system throughout the design cycle.

1.3 Related Work

Visualization is essentially a mapping process, taking data attributes and mapping them into visualizations [19, 33]. Data analysis systems are emerging that perform this mapping based on principles of perception and graphic design. Feiner [23] calls these systems "graphically articulate" because they attempt to convey the information in the data effectively and expressively.

Models and systems have recently been developed with this goal in mind [2, 14, 20, 43, 53, 60, 58, 70]. The applications vary in each of these research projects, from the presentation of charts and graphs to the presentation of images. However, they share a common goal to make the data analysis process more intuitive and productive. Researchers in this area have recognized the importance of the several disciplines necessary for the creation of effective graphical presentations, namely, computer graphics, data management, graphic design, perceptual psychology and user interface design. All of these researchers strive to define the components of the graphic design process, viewing current *ad hoc* methods as unacceptable for future graphics and visualization systems.

The project described in this thesis attempts to build upon the knowledge gained from these research projects. It expands on previous research projects by incorporating a structured framework that deals with data, user and machine characteristics in a single system. MDV takes into account these information sources as well as knowledge from perception and graphic design in producing a visualization. MDV also queries the scientist for his/her data analysis task. Task information is essential in order to design a visualization that meets the particular data analysis goal of the scientist.

The following subsections present a sampling of this work. The subsections are divided into four categories that represent a certain emphasis with respect to automated or assistant-based data analysis systems. Although each project addresses each of the focii to some degree, they are categorized by their main contribution toward the development of graphically articulate systems. Each project is summarized, followed by a brief statement of the lessons learned from the project and how the project differs from the current work presented in this thesis.

1.3.1 Focus on Conceptual Models

Haber and McNabb [30] presented the concept of a "visualization idiom", a geometric abstraction of the data obtained from a computer simulation. Designing an effective visualization idiom requires converting the raw data into a format that can be understood by the human visual system while maintaining the integrity of the data. Three transformations occur in the process of transforming raw data into a visualization idiom. The first transformation involves manipulating the data, performing operations such as subsetting, filtering, or smoothing. The next transformation involves selecting the visualization mapping. Choosing the correct mapping is important so that a clear interpretation can lead to a scientific conclusion. The final transformation renders the visualization mapping. All the information in the transformation process is necessary to understand the meaning of the resulting visualization idiom. Haber and McNabb state that idioms "should be based on intuitive analogies between familiar objects and the physical abstractions used in the simulation," so that understanding comes readily.

Haber and McNabb lay the foundation for the mapping process from data to visualization. In addition, they establish the need for selecting appropriate mappings based on the characteristics of the data. Their work is conceptual in nature and they do not go beyond the definition of a "visualization idiom" to develop an architecture or prototype system. Their emphasis is only on the data and they do not take into consideration the user and machine characteristics in the design of effective and expressive visualizations.

Robertson [53] stated the need for a methodology for guiding the choice of visualizations. He developed visualization guidelines by observing natural scenes in the world around us. Robertson asserts that human visual system contains visual mechanisms to understand our three-dimensional world, so it seems logical to visually represent data using physical properties found in natural scenes. Robertson's methodology matches data characteristics and data interpretation aims (data analysis task) to two-dimensional and three-dimensional scene parameters. This matching is performed by a human operator, but with potential for automated processing. His approach is different from other techniques in that he has a complete and coherent scene in mind before the mapping stage begins. By restricting the user to predefined scenes, the natural scene paradigm attempts to provide an appropriate mental model of the information displayed.

De Ferrari [54] expanded Robertson's original paradigm to include a more elaborate data model. In addition, she proposed the use of a "visualization specification" as input to the visualization system, which consisted of user directives and user interpretation aims. The goal of this work was a framework and a system that automatically generated visualizations.

Robertson's work also details the necessity for selecting appropriate visualizations based on the characteristics of the data in order for intuitive interpretation by the scientist. The natural scene paradigm is a methodology by which the mapping process can be determined. Robertson developed a prototype system based on the natural scene paradigm. De Ferrari's work was important in defining the need for an explicit data model and the user's goal-related input. Requirements for the data model were specified but a data model to support their automated system was not developed. In addition, neither Robertson nor De Ferrari cite the need to explicitly model user and machine characteristics in designing an effective visualization. However, the concepts and specifications developed in this work were very influential to research in the area of automated or assistant-based visualization tools.

Wehrend and Lewis [70] describe each visualization process by two dimensions: the attributes of the information to be displayed (*objects*) and the specific perceptual task to be performed (*operation*) on the resulting images. The finite number of data attributes and the finite number of perceptual tasks define a two-dimensional matrix in which each element contains expressive and effective examples of visualizations. If more than one tuple *(objects, operation)* is to be represented in the same display, the user is responsible for setting priorities for the composition of representations.

Wehrend and Lewis's work is a categorization of visualization techniques based on data attributes and perceptual tasks. This work was very valuable because it identified a subset of basic data attributes and perceptual tasks important for visual data analysis. In addition, it provided representative mappings for each *(objects, operation)* tuple. These representative mappings can be referenced when designing a visualization given data attribute (object) and task (operation) information. Wehrend and Lewis did not attempt to encode this knowledge in the development of a visualization system nor did they address user and machine characteristics.

1.3.2 Focus on Relational Information Display

Mackinlay [43] developed APT (A Presentation Tool) to automatically generate graphical presentations of relational information. The model is described by data, task, and user directives. Based on a description of the information and task, APT defines evaluation criteria and is able to compose multiple items and relations into one effective display. Mackinlay uses composition rules to define appropriate combinations of simple graphics primitives in the generation of representations. To map from an internal representation to displayable images, evaluation criteria such as importance (ranking of tasks), expressiveness (encoding of data attributes), and effectiveness (psychophysical principles) are used. Although restricted to relational data and two-dimensional charts, Mackinlay's work is a foundation for complex visualization systems aiming toward automating the design of graphical presentations. Mackinlay agrees that additional work is necessary in developing automated presentation tools for three-dimensional data, and automated tools for extracting and interpreting features in the data.

SAGE, developed by Roth and Mattis [58], is focused on the presentation of relational data using the 2D static displays found in business and statistical packages. Data characteristics are used in SAGE to create useful graphical displays. These characteristics describe the semantic[5] and structural properties of the data relevant to graphical design. For example, the nature of the ordering relationship among a dataset's elements is an important characteristics when choosing a graphical technique. Datasets may be categorized as quantitative[6], ordinal[7] or nominal[8]. Based on this data characterization, different graphical techniques are selected to convey the information in the data in the most appropriate manner. SAGE also addresses the role of application's

[5]Semantic properties are those that reflect the meaning of the data.

[6]Quantitative data is ordered numerically. Data from computer simulations is considered quantitative.

[7]Ordinal data is ordered by rank only. For example, the days of the week represent an ordinal dataset.

[8]Nominal data is unordered. For example, the different automobile manufacturers (*i.e.* Ford, Chevrolet, etc) represent a nominal dataset

or user's goals in designing effective graphical displays. The data analysis goal affects the effectiveness of a graphical technique.

Recent work by the developers of SAGE [57] has produced new paradigms for establishing collaboration between the user and the automated design system. The SAGEBRUSH system allows the user to create a rough sketch of a graphic design using a palette of primitives and partial designs. The system then improves on this design, taking into account semantic and syntactic qualities of the data. SAGEBOOK is an interface that provides users access to a catalog of previously designed images that can be applied to their data. SAGEBRUSH and SAGEBOOK are built on the foundation created by the original SAGE project.

Research performed by Mackinlay and Roth was intended for graphical presentations of relational information. Their systems were not oriented towards scientific visualization, but rather the development of charts and graphs. However, a great deal of overlap between relational information display and visualization occurs. Lessons to be learned from the work of Mackinlay and Roth include the representational techniques used to describe data as well as the methodology used in designing a visual representation. This can be applied directly to the development of automated or assistant-based tools for visualization.

1.3.3 Focus on Task Analysis

Casner's work [14] on BOZ establishes the necessity to incorporate the user's task in designing a graphic presentation, stating that the utility of any presentation is a function of the task it is being used to support. BOZ's task-analytic approach begins with a task and a description of logical operators that are necessary to perform the task. BOZ examines each logical operation and determines what information is required to perform the operation. The key step in BOZ's approach is replacing these logical operations with less effortful perceptual operations. By replacing logical operations with perceptual operations, the use of human information processing capabilities can be optimized.

Beshers and Feiner's work on the AutoVisual project [7] also deals with the graphic presentation of relational information. It is based on their earlier project, n-Vision [22], in which multivariate data can be analyzed in a hierarchical manner. The user specifies his/her visualization task and the system generates an interactive virtual world appropriate for the task. AutoVisual uses a two-part task specification composed of task operators and task selections. Task operators represent fundamental cognitive judgements to be made by the user, such as exploration, directed search or comparison. Task selections specify subsets of a relation and the data of interest.

Work by Casner and Beshers and Feiner is also geared toward relational information display. Lessons to be learned from their work include the importance of the task in designing a visualization. Their evaluation of task representation is valuable and transferable to scientific visualization.

1.3.4 Focus on Scientific Visualization

The VISualization Tool Assistant (VISTA) [61] is a system, developed by Senay and Ignatius, that designs visual representations automatically. VISTA emphasizes the mapping of data attributes to primitive visualization techniques, which encode one dependent and up to four independent variables. The synthesis of visualization mappings takes place in three steps. The first step involves decomposing the data so that each element can be represented by a single visualization primitive technique. The second step involves using rules of effectiveness and expressiveness to find the proper visualization technique for each data component. A search space of primitive visualization techniques is organized during this step. VISTA then searches this space until it finds a visualization technique that can express a given relation and that has not already been used to visualize another relation. In the final step, primitive techniques are combined to form a composite visualization by applying the appropriate composition rules. The user is able to interactively modify certain attributes of the visualization without causing inconsistencies in the final design.

Research by Senay and Ignatius has been very valuable, compiling an extensive collection of visualization-related rules as well as developing a methodology for visualization design. Their work does not address data, user and machine characteristics simultaneously in the design of visualizations. However, they discuss how the user and the task at hand are involved in designing visual representations [62]. Senay and Ignatius developed a prototype system, although the design time for creating a visualization using this prototype is often prohibitive. They provide little information on user feedback, so it is not known if the system has been tested by scientists.

Rogowitz and Treinish's [55] rule-based architecture contains visualization rules that the user invokes explicitly. Rather than building an autonomous system, Rogowitz and Treinish consider it a more feasible approach to allow the user to apply rules and consider the various paths, selecting those most relevant. These rules provide guidance on the selection of visualization parameters based on rules from human vision, cognition and color theory. If selected, the rules constrain the way in which the data is presented. The rule-based architecture contains a taxonomy of data structures that characterize features of the data and a taxonomy of visualization strategies that are based on principles of perception and cognition. The rules provide the link between the two taxonomies, designing visualizations matched to the capabilities of the user.

Rogowitz and Treinish's project requires the user to select and apply rules of perception to the design of visualizations. Their system does not select and render visualizations for the user, rather, it supplies the user with rules they can then apply to their situation. They note the need for a comprehensive data model that describes the data characteristics, but do not mention user and machine characteristics and how they influence the selection of visualizations. In addition, because the user is responsible for selecting the visualization mapping, task specification is not a component of their system.

Ahmed, Wanger and Kochevar [3] developed an Intelligent Visualization Subsystem (IVS) that uses a knowledge-based system to help users construct visual representations of Earth science data. A data model is a necessary component of the architecture, managing the heterogeneous datasets obtained from the application domains. The IVS knowledge-based component contains knowledge about the data model, the application domain, visual perception, and the characteristics of the output medium. The IVS uses information in the knowledge-base, together with a high-level task specification, to select a visualization mapping. The task specification requires the user to select an operator and a list of fiber bundle domains[9] upon which to operate. The initial three task operators incorporated into the system include select, search, and correlate. The task operator chosen is used to distinguish which visualization mapping is most appropriate. The knowledge base also contains information on how to build and render data-flow networks in AVS (Application Visualization System), once a visualization mapping is selected. Future work will involve incorporating information about the output medium and the user.

The IVS project is the most similar to the MDV project. Ahmed, Wanger and Kochevar's work, which is ongoing, notes the importance of data model in the design of a visualization assistant. The data model they have selected is the Fiber Bundle data model, which does not incorporate data semantics directly with the data. Rather, semantic information about the data is stored separately. This is different from the approach in this work, where semantic information is expressed in the data model together with the data it represents. The close association of data syntax, data semantics and data values in the MDV system provides a more intuitive representation of the data.

IVS developers also acknowledge the need to specify the data analysis

[9]The fiber bundle [12] model is used to represent the complex, geometrical relationships found in scientific data. For a further explanation of the fiber bundle model, see Section 3.1.1.

task for effective visualization. As a result, a query to the IVS system includes the specification of a data analysis task. The task operators used in the IVS (select, search and correlate) are different from the task operators in MDV (debug, explore, analyze and present.) The difference in task operators is due in part to the difference in domains. The IVS supports Earth science data analysis. The task operators in the IVS reflect typical tasks the Earth scientist might specify for their data analysis. In the MDV system the tasks are based on typical usage scenarios scientists working on CAS simulations might encounter. Although both sets of task operators could be applied to both applications, it is evident that each application domain comes with its own domain-dependent set of task operators. Thus, domain-dependent information is important in determining a task dialogue for a visualization assistant.

Finally, in comparison with the MDV project, the IVS project does not incorporate user or machine characteristics in the design of visualizations. However, the IVS project acknowledges the importance of these characteristics in designing a visualization and plan on incorporating user characteristics in the future. Information on user feedback is not provided, so it is not known if the IVS system has been used by scientists.

1.4 Summary

This thesis presents work in the development of a visualization assistant (MDV) for use in multidisciplinary computational aerosciences. MDV is a user tested system for assisting scientists with data analysis. This work is unique in that it establishes a framework for incorporating the several different information sources necessary for designing effective visualizations. The information sources include models of the data, the user, and the machine (hardware and software) environment. In addition, a knowledge base of visualization-related information is used in designing the visualization based on the data analysis task of the scientist. This framework can be applied to the development of a visualization system for any application domain.

Chapter 1 has provided an introduction to the thesis topic, describing the motivation behind the research, the goals and approach of the research and a summary of previous related work. The remainder of this thesis is organized as follows.

Chapter 2 describes the application domain that was selected for the design of the prototype visualization assistant. This chapter includes the data characteristics and visualization requirements of the application domain. Four user scenarios are then presented that demonstrate how CAS scientists perform their data analysis.

Chapter 3 describes the conceptual framework that is the basis for the visualization assistant. A special emphasis is placed on the data model, the main and most completely developed component of the framework.

Chapter 4 describes the prototype system, MDV (MultiDisciplinary Visualizer), developed based on the conceptual framework. This section describes the design process and the MDV interface.

Chapter 5 discusses user feedback about the MDV system. Conclusions and plans for future work are also presented in this section.

CHAPTER 2
APPLICATION ENVIRONMENT

Computational aerosciences, a multidisciplinary application in which the disciplines that affect the performance of aircraft are computationally modeled, was selected as the specific application for the design of the prototype visualization assistant, MDV (MultiDisciplinary Visualizer). This system was designed to assist researchers at the NASA Ames Research Center to visualize the results of computational aerosciences (specifically, fluid dynamics and structural dynamics) simulations.

This chapter details the application environment, describing the purpose and properties of computational aerosciences simulations. Computational fluid dynamics and computational structural dynamics, the two disciplines modeled in current simulation efforts, are then described in detail. Visualization for computational aerosciences is presented, describing the visualization mappings that are typical for the application domains. A detailed description of the simulation research in progress is presented, followed by usage scenarios, which describe how the scientist approaches CAS data analysis.

2.1 Computational Aerosciences

Designing safe and efficient next-generation high speed air transport requires the concurrent analysis of the several physical disciplines that affect aircraft performance. These disciplines include fluid dynamics, structural dynamics, thermodynamics, and propulsion. Forces from all of these disciplines affect aircraft performance. Fluid dynamics analyses provide information about the aerodynamic characteristics of the aircraft, such as the amount of lift and drag or the presence of vortices[10]. Structural dynamics analyses predict wing flutter and material failure. Thermodynamics analyses detect areas of extreme temperature that may affect aircraft integrity or passenger comfort. Propulsion analyses predict how engine characteristics affect the performance of the vehicle.

Currently, these individual disciplines are modeled separately, with the integration of results performed by humans. Single discipline simulations, such as those performed in computational fluid dynamics, require enormous computational resources. The amount of resources necessary to perform coupled, multiple-discipline studies is even greater. The advanced computing technology offered by parallel computer architectures has the promise of making multidisciplinary simulations feasible.

[10]Circulating regions of flow.

This advanced computing technology is very valuable to the aeronautics industry as traditional large-scale empirical tests (*i.e.* wind tunnel tests, flight tests) are not only costly, but often infeasible due to the high-altitude, high-speed flight conditions that must be reproduced. To design an optimized vehicle, coupled multidisciplinary computational models are necessary. Single discipline computer simulations have proven to be an effective tool in the design process, decreasing the design time cycle and improving aircraft efficiency, all at a reduced cost. Multidisciplinary simulations will undoubtedly follow this lead, playing an important role in future aircraft design.

The Computational Aerosciences (CAS) Project, part of the High Performance Computing and Communications Program (HPCCP) [32], investigates some of the difficulties that arise from incorporating several disciplines in a single simulation. One of the main goals of this project is a multidisciplinary analysis of a next-generation air transport, coupling fluid dynamics and structural dynamics. A basic description of these disciplines is presented in the following sections.

2.1.1 Computational Fluid Dynamics

Computational fluid dynamics (CFD), the analysis of fluid flow via numerical simulation, assists scientists and engineers in developing a better understanding of how airflow affects the flight characteristics of aircraft and aerospace vehicles. The CFD simulation process consists of grid generation, flow solution, and post-processing data analysis. Simulations can be performed for either two-dimensional or three-dimensional objects. However, for the rest of this thesis, I assume the simulations are of the three-dimensional variety. A discussion of post-processing data analysis for fluid dynamics and structural dynamics is deferred until Section 2.2.

Grid generation consists of creating a volumetric grid about the body of the aerospace vehicle, with nodes positioned in a Cartesian coordinate system. The most common grids are logically organized (or "structured") grids, possessing a connectivity that is implicitly defined by the three-dimensional arrays that contain the grid point positions. The physical positions of the grid points form curved and warped surfaces that conform to the geometry of the modeled aircraft. The grid cell sizes vary, from small cells near the vehicle to capture detailed physics, to large cells in the outlying regions of the grid where detail is not required. Typically, one or more computational grids exist to give more accurate detail to areas of complex geometry and physics. Figure 2.1 shows a simple CFD grid.

Flow solution involves solving the equations that govern fluid flow relative to a set of given boundary conditions. The basic laws of physics that apply

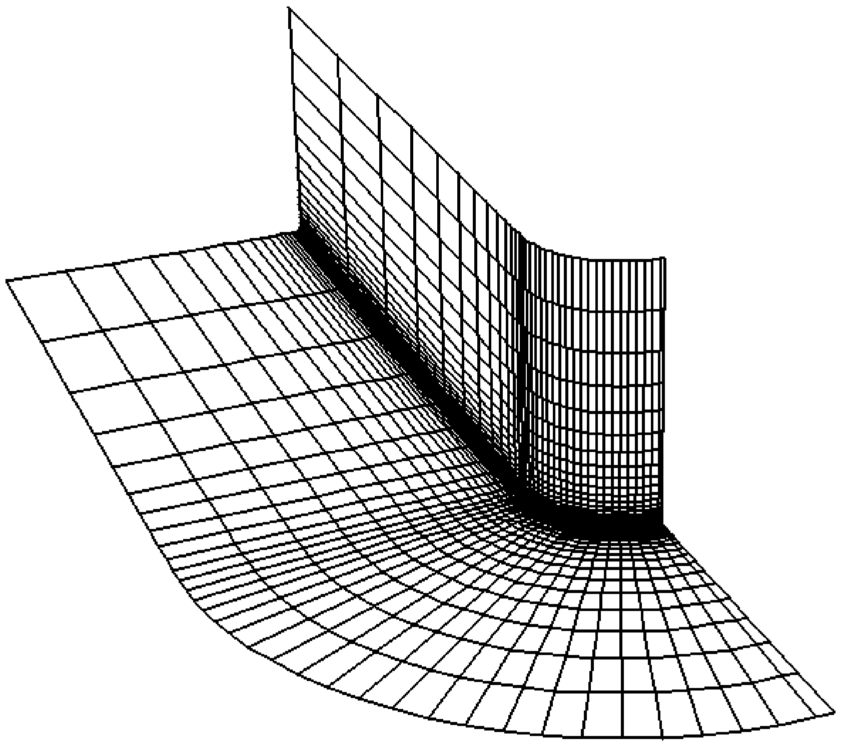


Figure 2.1: A simple CFD grid.

to fluid dynamics include the conservation of mass, conservation of momentum and conservation of energy. From these basic laws, equations are derived that are described by five field variables. These variables, computed at each node, are the density of the fluid, the x, y, and z components of the fluid momentum, and the energy of the fluid. All other field variables of interest can be derived from these five variables (*i.e.* pressure, velocity, etc).

The equations derived from the basic laws can be classified into three categories based on simplifying assumptions. The most basic equations are the potential equations that describe a fluid flow that is isentropic and irrotational. In simpler terms, isentropic flow is when no heat is added or taken away and where no frictional or dissipative effects occur, and irrotational flow is when vortices are non-existent. The Euler equations fall into the intermediate category, where the assumption is made that the fluid is inviscid. Inviscid flow is frictionless flow. Finally, the Navier-Stokes are the most complex and complete equations of fluid flow, taking into account the properties ignored in the above simplifications. The Navier-Stokes equations are most frequently used in aeronautics, describing flow by a set of nonlinear partial differential equations.

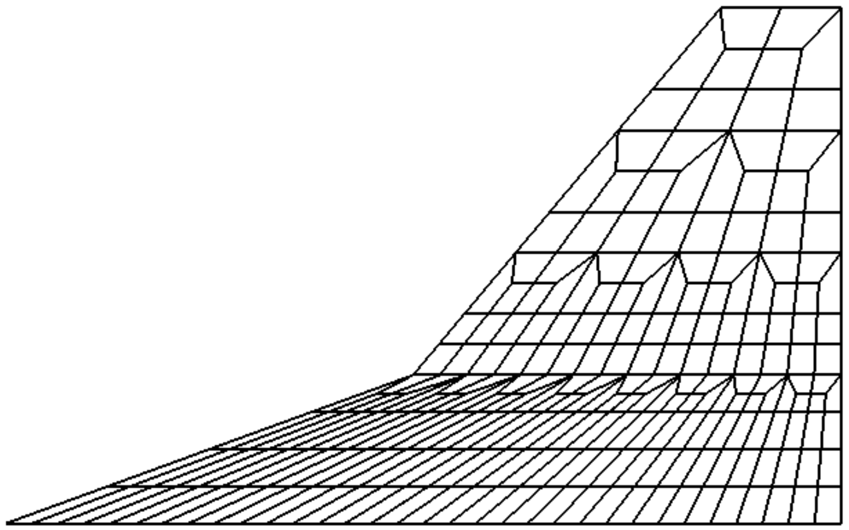


Figure 2.2: An unstructured computational structures grid.

2.1.2 Computational Structural Dynamics

The purpose of computational structural dynamics is to predict the behavior of a physical structure, such as an airplane wing, under actual operating conditions. Aeroelasticity, in which wing flutter results from fluid forces, is of particular interest to the aircraft community. The simulation process for computational structures is similar to that for CFD, consisting of grid generation, computation of the structural response, and post-processing data analysis.

The finite element method is typically used to formulate a model, or grid, of the object in the structures domain. The finite element method allows the dynamic behavior of a flexible body to be modeled as an assembly of specialized structural elements such as beams, plates, and shells. This formulation makes use of three-dimensional "unstructured" grids that describe the surface and interior composition of the object.

Unstructured grids do not possess the same rectangular connectivity as logically structured grids in CFD. Each element in an unstructured grid is defined by an arbitrary number of nodes and can be a variety of shapes (*i.e.* two-dimensional triangles and quadrilaterals; three-dimensional tetrahedra and prisms). Each grid cell is numbered, as is each node in the grid. The connectivity of the grid is explicitly specified by a list containing each element number and the node numbers that comprise that element. A simple unstructured grid, composed of quadrilaterals, is shown in figure 2.2.

Matrix methods are used in computational structures to analyze the dynamic and static responses of structures. Structural deformations are linear and governed by Navier's equilibrium equations. Properties such as the mass and stiffness of the surface material of the object are included in the computation of the object's response to external forces.

2.2 Visualization for Computational Aerosciences

Scientific visualization for CFD simulations has been one of the driving applications in the development of visualization software. The factors that have caused CFD applications to drive tool development are dataset size, complex physical phenomena, and the three-dimensional, time-dependent (temporal) nature of the simulations. Many visualization mappings are available to view the three-dimensional, time-dependent nature of fluid phenomena. Similar mappings are available for computational structures.

Several distinguishing data characteristics guide the selection of appropriate visualization mappings. The characteristics used in the development of the visualization assistant are described in the following sections.

2.2.1 Scalar, Vector or Tensor Fields

The most distinguishing characteristic of a data field is whether it is scalar, vector or tensor. Scalar fields are scalar quantities (of rank zero) defined on each node of the simulation grid. Scalar quantities include pressure, energy, density, and temperature. Vector fields, such as fluid velocity or momentum, are vector quantities (of rank one) that describe the direction and magnitude of a variable at each node in the field. Tensor fields, such as stress in computational structures, are tensor quantities (of rank two) defined at each node in the field.

Scalar field visualizations are the most common type of visualization generated in the computational aerosciences domain. There are several standard mappings to view scalar quantities. Colormapping consists of mapping data values to color values. This mapping is good for understanding trends and structure of the data; however, it does not convey quantitative knowledge well. Figure 2.3 shows pressure data colormapped on a fluids gridplane[11].

Contouring is an effective mapping for a quantitative analysis of scalar data. Scientists use contour mappings to easily compare and correlate data in the simulation. Contours show trends and offer greater numeric precision than

[11] A gridplane is a surface within a three-dimensional logically structured grid specified by holding one index is constant.

Figure 2.3: Scalar pressure data colormapped (top) and contoured (bottom) on a fluids gridplane.

colormapped datasets. Numerical values are often displayed with corresponding contour lines; however, this causes visual clutter. Color-coding contour lines or selecting the contour line with the mouse to display the value can alleviate the problem of clutter. Figure 2.3 shows contour lines portraying the pressure on a fluids gridplane with a popup menu containing the data value.

Isosurfaces are the three-dimensional extension of contour lines. Isosurfaces are surfaces of a single scalar value in a three-dimensional field. They are helpful in understanding structure by isolating data of a constant value. However, because isosurfaces of different values obscure each other, only a few can be viewed simultaneously. Techniques such as transparency can help to address this problem.

Vector field visualization requires mappings that are able to capture both the magnitude and the direction of the data. A simple mapping involves the drawing of small vector arrows for each grid point in the data. This mapping, called an arrow plot (sometimes referred to as a vector plot or hedgehog), gives the user a feel for the directionality of the flow field by viewing the texture. Although this mapping is simple in form, it can be confusing to interpret by the scientist when the density of data points is high. The collection of arrows can become cluttered, causing confusion, especially when a three-dimensional subset of the grid is depicted using an arrow plot. As a result, the scientist must select a relatively small portion of the flow field with a density of arrows that may not be sufficient to show fluctuations in the flow field. Velocity arrow plots are shown in figure 2.4.

Another mapping for displaying vector fields is the streamline. Streamlines are curves that are tangent to the flow field at an instant in time. In a steady flow (where the flow field does not change with time) the streamline represents the path a massless particle would take in the velocity vector field. Scientists must specify the initial seed location of the particle. The placement of the seed location is often done by trial and error, searching the field for regions of interest.

Because a single streamline represents only a single particle, it is difficult to obtain a more global impression of the flow field. Scientists often use a collection of particle seed locations located close together along a line. This set of seed locations is referred to as a "rake," and it generates a family of streamlines as shown in figure 2.4. This visualization gives a broader representation of the vector field.

In complicated flow regions, for example, where vortices are located, streamlines generated from a rake can become twisted and entangled, which can be visually confusing, since the streamlines are no longer visually distinct. Streamlines are often given a thickness so that they become thin ribbons. This mapping allows the scientist to better distinguish the individual particle paths

Figure 2.4: Vector velocity depicted by an arrow plot on a fluids gridplane (top) and by a family of streamlines (bottom).

and judge their curvature in the flow field.

However, ribbons can still be difficult to discern and the scientist is left with a complex image of the flow field. In addition, the ribbons twist only in the streamwise direction of the flow. The stream surface, shown in figure 2.5, is a mapping to view the vector field as a surface so that the curvature of the flow can be seen downstream and across the width of the surface [34]. The surfaces bend and twist with the flow, generating an image that is useful in depicting complex fields.

Visualization mappings to view tensor quantities are still highly speculative and in very early phases of research [17]. As a result, they have been omitted from this work.

2.2.2 Spatial versus Temporal

Visualization mappings may be spatial or temporal in nature. Spatial mappings do not take into account the time variable when constructing a visualization. Temporal mappings, in addition to spatial information, are distinguished by their use of the time variable when constructing a visualization.

Colormapping is an example of a spatial mapping which is applied to a single timestep of data, such as shown in figure 2.3. Mappings such as streamlines and stream surfaces are also spatial, using only the information contained in a single timestep. Pathlines and streaklines are two temporal visualization mappings that are applied to multiple timestep datasets [41]. A pathline shows one trajectory of a particle released at a given moment in time from a seed point. It is interesting to note that without the time variable, the pathline and streamline are identical. A streakline is the locus of an infinite collection of particles that have been continuously released from a single location. The use of temporal mappings such as pathlines and streaklines provide the scientist with a more intuitive feel for the progression of flow in a dynamic flow field.

Animation is an important technique for viewing related spatial mappings. Related spatial mappings may consist of different views of a time-independent dataset, or a sequence of spatial mappings applied over a time-dependent dataset. Animation is used to view the results of temporal mappings. The use of animation reinforces the time-dependent nature of the data. However, not all temporal mappings need to be animated. For example, a simple line plot of the time history of a data point may provide a clear interpretation of the dataset dynamics.

Figure 2.5: A complex fluids velocity field depicted by entangled streamlines (top). The same flow is depicted by a stream surface, simplifying the representation and showing the curvature of the flow field.

2.2.3 Continuous versus Discrete

Most of the phenomena modeled in computational aerosciences are continuous in nature. The discrete nature of the computational grids is typically reconstructed into a continuous space using interpolation methods. Mappings such as contours, color contours, surfaces, streamlines, and streamsurfaces provide good methods for viewing continuous data fields.

However, certain variables are computed at the nodes of the computational grids and are discrete values. When continuous mappings are applied to data of discrete type, a visualization can create misleading artifacts. An example of discrete data is the structures variable *nodal force*. *Nodal force* is computed at each node of a finite element representation, based on pressure integrated over the area of adjacent elements. A colormapped visualization, shown in figure 2.6, is a misleading representation of the data. The visualization implies that the nodal force values are distributed evenly over the surface of the finite element wing. However, this is not the case, since the nodal forces are intended to be discrete data values, computed at each node on the wing. A better representation of this variable would be to use a discrete mapping, such as a collection of discrete symbols, with each symbol colored with the magnitude of the force (shown in figure 2.6).

2.2.4 Local versus Global

Visualization of CFD data involves working with three-dimensional grids that are difficult to portray on a two-dimensional medium such as the computer screen. Scientists often choose to view small, local subsets (one-dimensional or two-dimensional) of their data using mappings such as line plots (one-dimensional), colormaps or contours (two-dimensional). These local representations are effective in examining detailed sections of the dataset to be analyzed. The scientist typically knows where the regions of interest are located, so these mappings are sufficient.

Data subsetting is required for viewing local mappings such as colormappings, contouring and vector plots to avoid visual clutter and confusion. Data subsetting can involve, among other options, selecting a gridplane of the computational grid, selecting an arbitrary cutting plane that is some cross section of the computational grid, or selecting a subvolume of the grid.

When the scientist is not familiar with the general structure of the data, he/she may choose to view the entire three-dimensional, global view of the data. Visualization mappings exist that can allow the scientist to portray the entire dataset at once for both scalar and vector data fields.

For scalar data, such as pressure, temperature or local Mach number, volume rendering can be an effective way to view the overall structure of the

Figure 2.6: Nodal force visualized using a continuous mapping (top) and a discrete mapping (bottom).

data. One method for volume rendering is based on a ray-casting algorithm. The ray traverses through the volume, accumulating color and opacity information obtained by integrating a function of the data it encounters [69]. Vector data can also be visualized using this mapping, given a function that can map the data expressively for interpretation by the scientist.

A mapping to display a complete three-dimensional vector dataset is vector field topology [27]. Vector field topology is a mathematical analysis of the vector field that consists of finding points, curves and surfaces that characterize the integral manifolds in the vector field. The integral manifolds include particle traces, streamlines, and stream surfaces. By visually analyzing this mathematical representation of the flow field, the scientist can locate points of interest such as critical points (points of zero velocity), vortex cores (the theoretical "center" of a vortex) and other structural landmarks in the flow field. Using these features as guidepoints, one can then use local mappings for further study of the data.

2.3 The HSCT Grand Challenge Problem

One of the grand challenge problems[12] under investigation at the NASA Ames Research Center is the High Speed Civil Transport (HSCT). Scientists are developing multidisciplinary simulation software, coupling fluid dynamics and structural dynamics [50], in order to examine the flight characteristics of supersonic transports. The HSCT is a government-sponsored research project to design and develop a new high-speed commercial jet by the year 2005. The range of the HSCT is 5,000 nautical miles and its capacity is 300 passengers. A gridded surface of a model of the HSCT is shown in figure 2.7.

2.3.1 Simulation Overview

The simulation, currently modeling only a wing of the HSCT, is performed using a direct coupling of the physical disciplines to study the aeroelastic effects of aircraft flying near the speed of sound. Because of the computational power required for multidisciplinary simulations, the use of parallel computers is under investigation. For this simulation, a 128 processor Intel iPSC/860 was used [50]. The Mach number[13] selected for the computation was 0.7 with an angle of attack[14] of one degree.

[12]Grand challenge problems are large-scale computational problems whose solution would impact progress in science and engineering. The computational problems associated with the HPCCP CAS project are described as grand challenge problems.

[13]The speed of the aircraft, divided by the speed of sound.

[14]The angle of the wing with respect to the direction of the airflow.

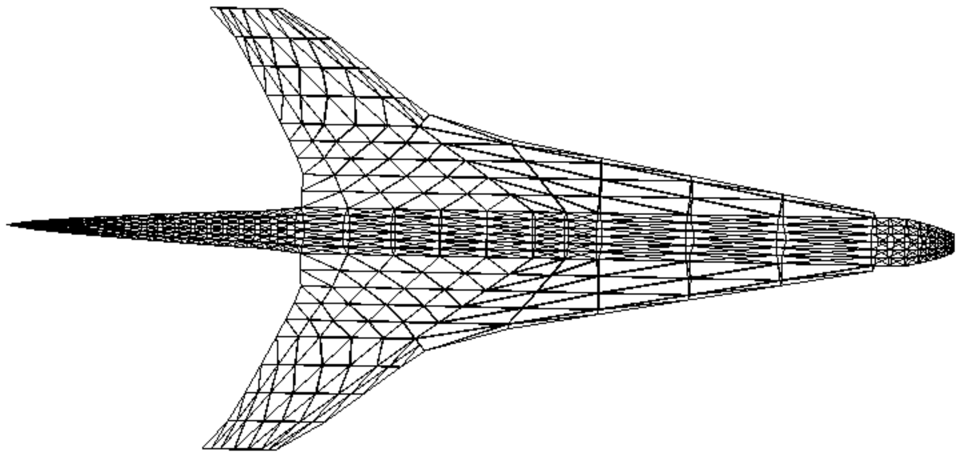


Figure 2.7: The High Speed Civil Transport (HSCT).

The direct-coupled, fluid structure interaction methodology involves 1) a numerical scheme for time integration of the flow field, 2) a numerical scheme for time integration of the structural velocity field[15], 3) a grid update scheme that adjusts the fluid grid based on structural deformations, and 4) a time stepping mechanism that couples the three components efficiently and accurately. Data transfer is performed at the interface of the two disciplines, specifically at the surface of the wing.

In the fluids domain, the flow field about the HSCT wing is discretized using a structured, body-fitted, curvilinear grid. The density of the grid varies spatially, based on the desired resolution in the various regions of the flow field. The varying grid cell sizes are intended to conserve storage and computation, while still capturing the detailed physical phenomena that occurs in regions of interest such as in and near the boundary layer[16]. The fluids computation was performed on a grid with 184,975 grid points using Euler's equations for inviscid flow. Two gridplanes of the three-dimensional structured fluids grid are shown in figure 2.8.

In the structures domain, the wing was discretized using an unstructured, finite element grid. The discretization of the two domains do not coincide at the surface of the wing; each domain has its own discretization requirements. The wing was modeled using a coarse irregular grid of 921 nodes. The wing is shown in figure 2.9. The top half of the figure shows the configuration "pulled apart"; separated into its upper surface, lower surface and internal structure. The bottom half shows the complete configuration in proper alignment.

[15]Structural velocity represents the rate of change of the deforming grid.

[16]The boundary layer is the thin layer of fluid near the surface of a modeled geometry, such as the HSCT wing.

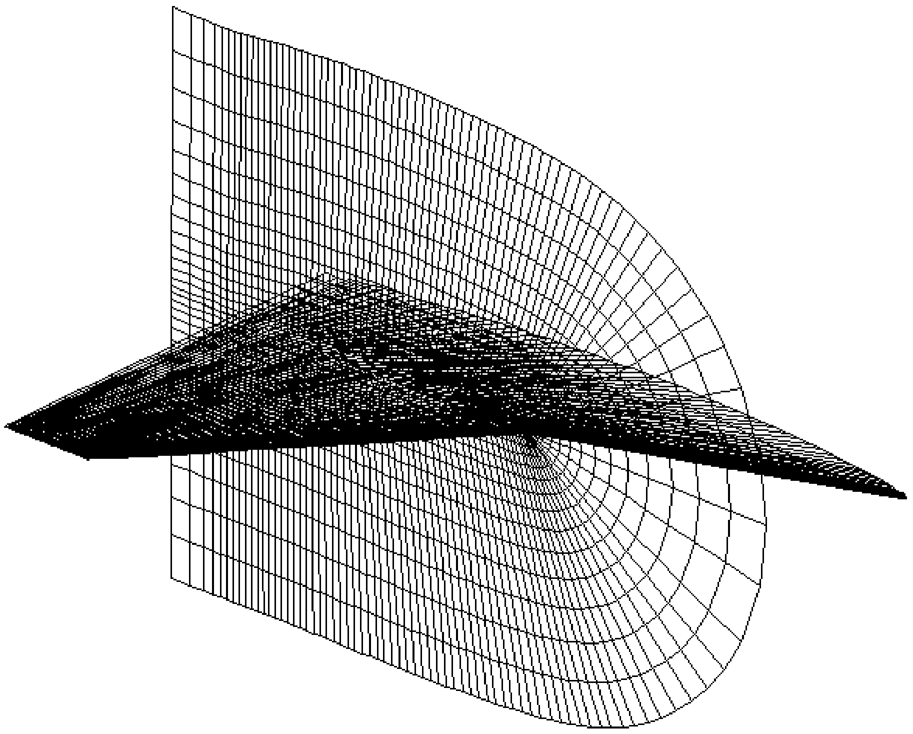


Figure 2.8: The HSCT wing fluids grid.

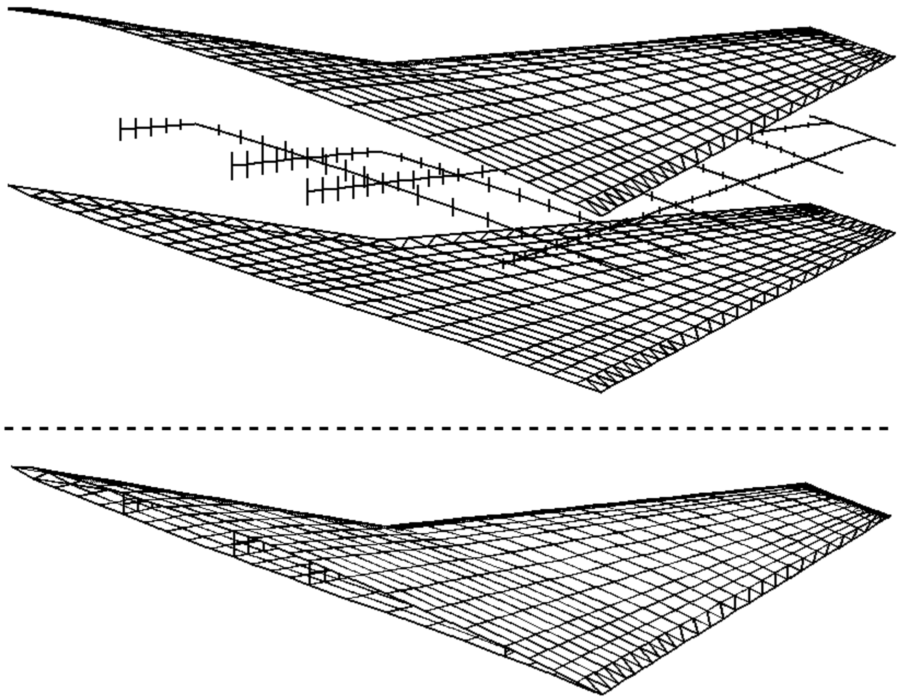


Figure 2.9: The HSCT wing structures grid.

2.3.2 User Scenarios

As a first step in the design of the visualization assistant, scientists were consulted as to how they typically approach their data analysis. User scenarios were created based on this information which helped to shape the development of the MDV visualization system. The design of these scenarios provided information about the typical tasks scientists were interested in performing. In addition, scenarios gave valuable information as to how to design the interface by establishing what information needed to be included in the system and in what order. The resulting user interface, which will be presented in section 4.2.1, was designed to accommodate the following sample scenarios used when analyzing computational aerosciences data.

Each of the data analysis scenarios is presented in four steps. First, a description of the data analysis task is presented. Second, a high-level, natural language query is distilled from the task description. Third, this query is broken down further into simple components that can be easily implemented in a menu-based system. Finally, the resulting visualization and how it helps to accomplish the data analysis task is described.

2.3.2.1 Scenario One: Debugging

The scientists involved in performing a multidisciplinary simulation are developing their simulation software. One of the major components of the code automatically updates the computational grids so that both correctly depict the deforming wing of the HSCT. However, because of the complex nature of the structural deformation, the alignment of the grids does not match. The scientists are interested in viewing where the mismatch occurs and how it progresses, which helps them to adjust their algorithm accordingly. In order to accomplish this task, the scientist generates the following query:

> Find the mismatch error between the fluids and structures grid positions on the HSCT wing.

This can be rewritten using the following basic query parameters. These parameters were common amongst all scenarios. These parameters became the basis for the user interface design.

1. Dataset Name: The HSCT dataset,
2. Fluid Object/Structures Object: The wing of the HSCT, in both cases,
3. Fluid Variable/Structures Variable: Position, in both cases,

4. OUTPUT: The color printer, so that this visualization may be printed for this thesis (the default value for this parameter is the computer screen),

5. TASK: Debugging.

Because the scientists are interested in viewing the fluids grid, they must select if they want to visualize all of the grid or only a subset of the grid. This information is not known by the system and must be entered by the scientists.

After selecting the grid subset, the visualization is designed and rendered first to the screen. Although the visualization is rendered to the screen, it is made suitable for printing to a color printer. The scientists only need to click on the image and send the output to the color printer. The resulting image is shown in figure 2.10.

This image shows the mismatch of the two grids at the tip of the HSCT wing. Although a single timestep of the visualization shows the error, the animation of all the timesteps shows the progression of the error. This aspect of MDV was very helpful to the multidisciplinary scientists in the early stages of their code development.

2.3.2.2 Scenario Two: Exploration

The scientists are interested in viewing the relationship between stress on the wing and the corresponding velocity field of the fluid. This relationship is helpful in understanding the dynamics of aeroelastic wing fluctuations. In order to accomplish this task, the scientist generates the following query:

> Explore the relationship between the fluids velocity field and the stress variations on the HSCT wing.

This can be rewritten using the following basic query parameters.

1. DATASET NAME: The HSCT dataset,

2. FLUID OBJECT/STRUCTURES OBJECT: The wing of the HSCT, in both cases,

3. FLUID VARIABLE/STRUCTURES VARIABLE: The fluids variable velocity and the structures variable stress,

4. OUTPUT: The color printer,

5. TASK: Exploration.

Figure 2.10: Scenario 1: Debugging the simulation software.

The result of this query produces a colormapped animation of stress on the deforming wing, with velocity streamlines about the wing. The resulting image is shown in figure 2.11. The interaction widget, shown in red, allows the user to explore the dataset by "mousing around" and moving the origin of the family of streamlines to probe areas of interest.

This interaction mechanism provides a more intuitive way to explore the relationships between the flow field and geometry. Other visualization systems, such as FAST, provide similar visualization mappings, but the user is required to specify the location of the seed points of the streamlines by indirectly using text or widgets. The direct manipulation mechanism is much more effective for exploration.

2.3.2.3 Scenario Three: Analysis

The scientists are interested in viewing the changing pressure forces on the wing and the resulting stress from those forces. They want to verify that the data is evolving in accordance with experimental tests. This analysis task is geared toward combining quantitative analysis with qualitative visualizations. This goal is accomplished by querying data values and plotting their time-histories. The data values are queried by picking points on the visualization with the mouse. In order to accomplish this task, the scientist generates the following query:

> Quantitatively analyze the relationship between the fluids pressure field and the stress on the HSCT wing.

This can be rewritten using the following basic query parameters.

1. Dataset Name: The HSCT dataset,
2. Fluid Object/Structures Object: The wing of the HSCT, in both cases,
3. Fluid Variable/Structures Variable: The fluids variable pressure and the structures variable stress,
4. Output: The color printer,
5. Task: Analysis.

Again, because the scientists want to study the fluids grid, the system asks if they are interested in the entire grid or a subset of the grid. In response to the query, the system designs a colormapped animation of stress viewed on

Figure 2.11: Scenario 2: Exploring a Velocity Vector Field.

the deforming wing. Fluid pressure is shown on the selected gridplane using contour lines. The resulting image is shown in figure 2.12. The user may click the mouse on data points of interest to query their value. The scientist can choose to plot the data values over time as shown in the visualization. The small red cross-hair denotes the node selected for plotting.

This visualization gives the scientist a mechanism for comparing data values with experimental data. The plotting capability shows the scientist if the oscillation of the wing acts as predicted. For this case, the oscillation is transient[17] as expected.

MDV selected not to represent both field variables using colormapping techniques. This decision was made to avoid confusion when interpreting relationships between data variables. Because the field variables and their respective dynamic ranges were different, incorrect relationships might have been established had both data values been colormapped. In order to avoid this confusion, colormapping was used to depict one field variable (stress) while contouring was used to depict the other field variable (pressure).

2.3.2.4 Scenario Four: Presentation

The scientists are interested in preparing a presentation graphic for a paper they are submitting to a conference. They are interested in showing the pressure variations in the fluids domain together with the deforming grid of the structures. Because of cost constraints, the image must be black and white. In order to accomplish this task, the scientist generates the following query:

> Generate an image for a conference that shows the relationship between the fluids pressure field and the structures position of the HSCT wing.

This can be rewritten using the following basic query parameters.

1. Dataset Name: The HSCT dataset,
2. Fluid Object/Structures Object: The wing of the HSCT, in both cases,
3. Fluid Variable/Structures Variable: The fluids variable pressure and the structures variable position,
4. Output: A black and white laser printer,

[17] An oscillation is transient if its magnitude diminishes over time.

Figure 2.12: Scenario 3: Analysis of fluid/structure simulation by combining quantitative analysis with qualitative images.

5. Task: Presentation.

The scientists are asked to select a subset of the structured fluids grid. A basic visualization mapping is selected because the resulting image is intended for presentation. A contour plot of the pressure field is presented which shows the varying pressure field and a clear representation of the structures wing. The resulting image is shown in figure 2.13 and is typical of what might be found in journal or conference publications in the scientist's domains.

These four scenarios were identified as four of the main types of data analysis tasks multidisciplinary scientists are interested in performing on their data. These tasks are only four of the many possible tasks that the scientist may be interested in performing. However, for the purposes of this project these tasks were the only ones studied.

2.4 Summary

This chapter presented an overview of the application environment selected for the development of the visualization assistant. This environment, computational aerosciences, is complex and unwieldy for scientists interested in performing data analysis. As a result, this application environment provides the motivation for the development of a visualization assistant. Although the development of a visualization assistant can be applied to any application environment, this particular environment was a good choice for the development of the prototype system for several reasons. First, scientists are unfamiliar with other scientific domains. This increases the need for a visualization assistant to assist them in analyzing data from an unfamiliar domain. Second, the large number of data files and complexity of the different data types and data formats demonstrates the benefit of a more automated approach. Finally, because the scientists didn't have a visualization system that could show both domains simultaneously, they were enthusiastic about participating in the design of MDV.

The user scenarios developed when working with scientists proved to be a valuable experience. By understanding the major classes of tasks the scientists are interested in performing, the design and development of the user interface became clearer. The scenarios provided the necessary structure from which to build the outline of the interface. As the system becomes more sophisticated, additional scenarios can be incorporated to provide additional functionality. This method of interface design is highly recommended.

The next chapter presents the underlying framework for the visualization assistant. The next chapter discusses how the data characteristics, domain knowledge and usage scenarios were incorporated into the visualization framework.

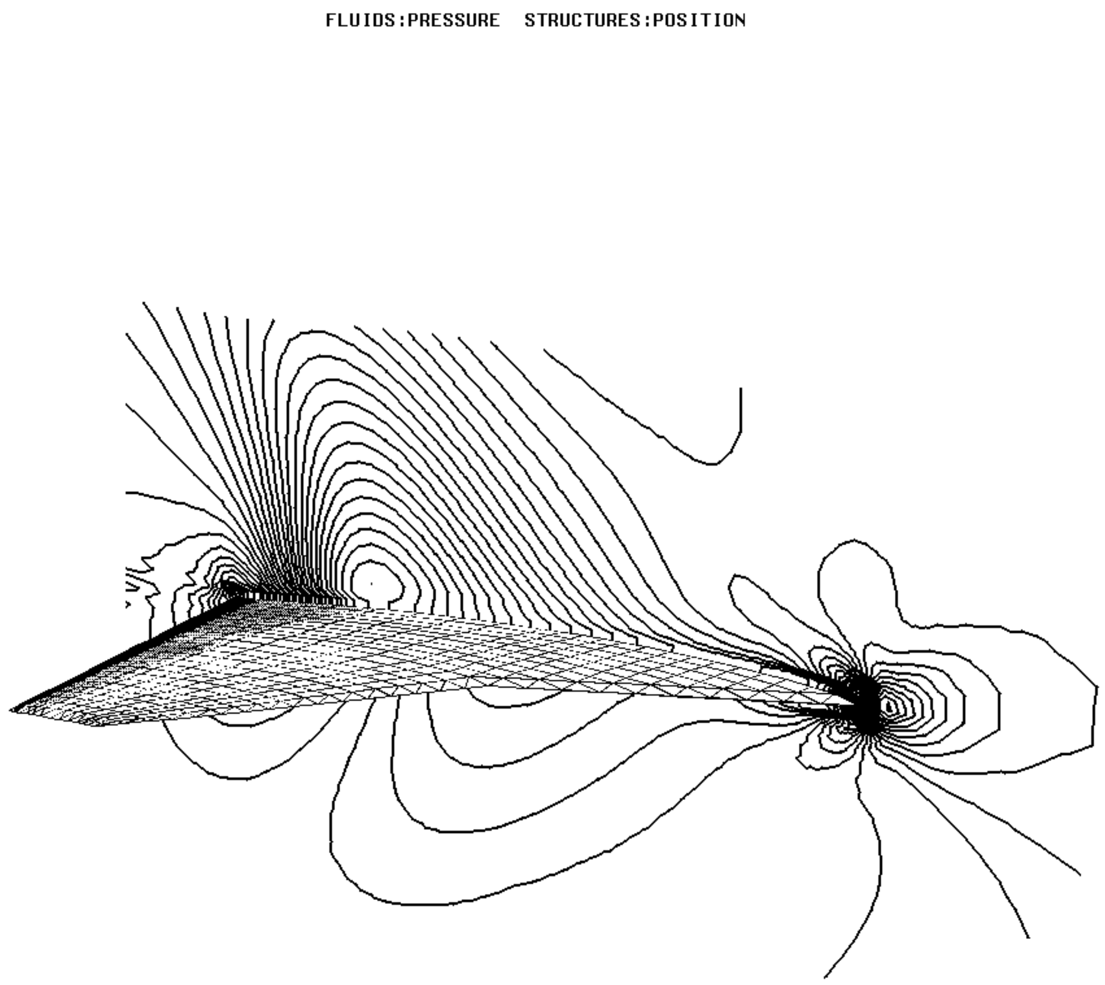


Figure 2.13: Scenario 4: Presentation graphic of fluids pressure on the HSCT wing, using black and white for printing constraints.

CHAPTER 3
THE VISUALIZATION FRAMEWORK

This chapter presents the framework for the visualization assistant. The framework consists of a data model, a user model, a machine model and a knowledge base. The three models were chosen because they each characterize an essential information source that impacts the design of a visualization. The knowledge base contains additional information that supports the visualization design process.

Each information model is described individually in the next three sections. In each section, a general definition and purpose of the model is presented. This definition is followed by a description of the contents of the model and how it is implemented in the visualization assistant. The model description also identifies the attributes contained in the model that influence the design of visualizations. These sections are followed by a section describing the knowledge base and its content. Finally, the process of designing a visualization based on the information models, the knowledge base, and a given user directive, is described.

3.1 The Data Model

The data model is the main and driving component of the visualization framework. The following section defines the concept and properties of a data model. Also included in this section is a classification of the different types of data models used in visualization systems.

3.1.1 Definition and Purpose

Recent interest in data modeling for scientific visualization has been demonstrated [5]. A data model is a mechanism for managing data so that data and information about the data can be retrieved easily and quickly. The data models described in this section represent the current use of data modeling concepts in the field of visualization.

The incorporation of database ideas into visualization systems is still in early stages and many avenues have yet to be explored. As is common when combining work from different domains such as visualization and databases, terminology and basic conventions may not be consistent. For example, in the visualization domain, the term "data model" is used to describe a physical and/or logical representation of scientific data. In contrast, the term "data model" in the database domain is used to describe an abstract modeling paradigm (*i.e.* the object-oriented data model, the relational data model).

Visualization data models are typically crafted using database data modeling paradigms. The term "data schema", in the database domain, is used when a representation of the data is specified using a data modeling paradigm. This term is probably a more precise term when describing specific visualization data models. However, visualization researchers have blurred the distinction between the terms and "data model" has become the standard for both conceptual database models and their resulting implementations. The terminology in this thesis reflects the use of data modeling terminology in the visualization field.

In the visualization community, two levels of abstraction are commonly used when defining data models. At the more basic level, data is described in terms of their physical layout or data format. At the more abstract level, data can be described by their logical layout and semantic content. Physical data models are representations of data on a physical level, representing how data is stored in the computer. In this case, the logical structure of the data model is closely tied to the physical representation of the data. These models are very effective for standardization and interchange of data and often serve as the internal data model for visualization systems. However, physical data models do not possess enough high-level abstraction nor do they contain the mechanisms for modeling the information necessary for mapping data effectively to visualizations.

Some of the more successful visualization-oriented physical data models include CDF [65], netCDF [52], and HDF [48]. NASA's CDF (Common Data Format) was designed to provide support for multidimensional, block-structured data. Unidata's netCDF [52] was based on the CDF concept of multidimensional blocks of data, but with a specific emphasis on issues of data transport. HDF (Hierarchical Data Format) [48], from NCSA, was designed to meet the need of data transport among heterogeneous machines. This need stemmed from the requirement to access images and other data generated on supercomputers from personal computers and workstations.

Conceptual data models are high-level data models, providing abstractions to structure the data similarly to how the user perceives the data. These models are high-level in their specification, and their approach allows a great deal of flexibility. Conceptual data models can fall into two categories, general purpose and special purpose. General purpose conceptual data models are specified using the true conceptual data models that are found in the database field. Some of the more common database modeling paradigms include the relational, extended-relational, semantic, and object-oriented.

The object-oriented and extended-relational data models are most commonly used when specifying scientific data. They are superior to traditional hierarchical, network, and relational models because they can manage the multidimensional, heterogeneous structure of scientific data. Concepts such as a class hierarchy, inheritance, and methods are useful for modeling scientific data and are not present in the traditional data models. In addition, the traditional models provide little support for the semantic information that is useful in describing scientific data.

For their visualization system, Sabella and Carlbom [59] used the object-oriented model to represent empirical data from oil reservoir studies. They used the object-oriented paradigm to manage the many different types of geometric and non-geometric data that are used in reservoir analyses, such as three-dimensional solid bodies combined with surface data, two-dimensional image data, and one-dimensional curves. In this case, the object-oriented model provided a useful abstraction mechanism, supporting complex query operations on the heterogeneous data.

A special purpose data model is one developed specifically with a certain type of data in mind. For example, the fiber bundle model [12] was developed to suit the needs of scientific data [18]. Fiber bundles are based on differential geometry and provide a high-level representation of many types of geometrical objects found in scientific data. Basically, a fiber bundle is specified by two spaces, a base space and a fiber space. Elements from the fiber space are attached to each point of the base space. The result may be thought of as a generalization of a function of one or more variables, in which the base space is analogous to the independent variable and the fiber space is analogous to the dependent variables.

Ahmed, Wanger, and Kochevar use the fiber bundle conceptual data model in the implementation of their visualization system [3], as described in section 1.3.4.

3.1.2 Implementation

The object-oriented data modeling paradigm was selected for the MDV visualization assistant because of its conceptual modeling mechanisms and straightforward implementation route. The concepts and modeling mechanisms of the object-oriented paradigm are presented in the following section. This basic description of the object-oriented approach is intended to familiarize the reader with the properties of the object-oriented paradigm. This description is followed by several reasons that demonstrate why the object-oriented paradigm a wise choice for managing computational aerosciences data.

[18]If desired, the fiber bundle model can be applied to many other types of data such as relational data.

3.1.2.1 The Object-Oriented Paradigm

The main concepts of the object-oriented paradigm include abstraction, encapsulation, and inheritance [10]. Abstraction is an effective way to deal with complexity, breaking down detailed systems into simple conceptual objects with distinct boundaries. An abstraction describes all the necessary characteristics of an object that make it a distinct entity to the viewer.

Encapsulation captures the essence of the object paradigm, providing a clear separation between the external workings of objects and their internal implementation. Encapsulation and abstraction are complimentary to each other. Abstraction focuses on the outside view of the object, while encapsulation provides a mechanism for hiding the inside, detailed view, which does not need to be seen by the viewer.

Although abstraction allows for the decomposition of complex systems, there is usually more information than the viewer can handle. Abstraction is often not enough to manage the complexity. An ordering mechanism must be incorporated to organize these abstractions so that the viewer may comprehend the overall system. Inheritance is one mechanism that performs this ordering. Inheritance defines relationships among objects. A class of objects can inherit structure or behavior from another class of objects.

Basic modeling concepts are necessary to specify an object-oriented model. The following core concepts are generally used by the members of the object-oriented community[19] [40].

- *Object:* An object represents a real world entity.
- *Attributes:* The values of the attributes of an object constitute the state of the object.
- *Methods:* The set of methods associated with an object define behavior, operating on the state of the object.
- *Classes:* Classes organize collections of objects with similar attributes and methods. Each object is an instance of a class.
- *Class Hierarchy and Inheritance:* A class may have any number of subclasses that inherit properties from this superclass. The organization of classes is the class hierarchy.

Metadata is a concept associated with data modeling. Metadata can most easily be described as data about data. It can take on many forms, from

[19]The Object Database Standard [15] has been developed as an attempt to standardize object-oriented terminology. However, this standard is not yet widely accepted.

describing the structure of the data, to codifying knowledge about the data. Basically, anything that establishes a context for the primary data (in this thesis, the data computed by scientists) can be considered metadata.

The concepts of metadata and attributes (as described above) are often used interchangeably. Attributes are typically considered a subset of the broader metadata class. Attributes are generally used to describe the syntactic, structural information about the data, including its type, dimensions, and rank. Metadata contains additional semantic information of the data, such as the names and relationships of the physical variables, contextual data about the simulation, and the processing history of the data. It is often the case that the term "attribute" is used in a general sense to describe a specific piece of information about the data, whether it is syntactic or semantic. In this thesis, the term "attribute" is used in this general sense and describes both the semantic and syntactic properties of the data.

The high-level modeling mechanisms of the object-oriented model permit the creation of a general model that is a flexible foundation for all disciplines in a multidisciplinary setting. The uniform representation of data across disciplines insures that data complexities are encapsulated, and that the functionality of the visualization system can be applied to all disciplines involved.

The object-oriented model was selected over other modeling paradigms because its modeling capability is superior when handling hierarchical, multidimensional, scientific data. For example, it is difficult to handle the simple interrelationships found in a two-dimensional array using the relational model [5]. In addition, capturing semantic information about the data using the relational database model is difficult. The fiber bundle model does not contain mechanisms for handling semantic information. Semantic information is necessary in the design of the visualization assistant for describing the data attributes that influence the design of visualizations.

In addition to its modeling capability, the object-oriented model possesses several properties that make it a wise choice for multidisciplinary scientific datasets. Methods, or operations defined on data objects, have the property of being polymorphic. That is, these methods can be applied to objects of different type, which allows for code re-use. For example, the dynamic range of a field variable can be computed appropriately if it is a vector field or a scalar field. In addition, only one method is necessary for computing contour lines regardless if the grid that describes the data is structured or unstructured. The ability to abstract the complexities of the data allows for a software environment that is easy to comprehend, expand and maintain.

The capability to derive data when queried allows the model to express the information content of the data without having to explicitly precompute and store all data variables. Methods defined on the data objects "know" how to compute any field variable queried by the scientist. The result is a semantically rich data model that characterizes the essential information needed by the scientist for data analysis.

The extensibility of the object-oriented model is beneficial so that additional disciplines can be added to the multidisciplinary application as necessary. Because the data model is high-level in nature, additional disciplines can be added easily by the application developer simply by modifying the model (to be described in section 3.1.2.2) for the additional discipline. For example, if thermal data is integrated into the data model, an object representing the thermal dataset would be generated and added to the application. Additional methods and attributes would be defined to reflect the specific properties of data values in the new discipline.

3.1.2.2 A Generalized Data Model for CAS Data

A generalized data model[20] for time-dependent data (shown in figure 3.1) has been designed, using the object-oriented modeling paradigm, to manage the data and information associated with computational aerosciences simulations. The data from each domain simulated in a multidisciplinary computation is modeled as a specialization of this general model. As part of the specialization, domain knowledge is embedded into each of the domain-dependent models in the form of attributes. Attributes that influence the design of visualization mappings are also incorporated into each of the models.

This model was developed after several iterations on its design. The design process involved studying the data management requirements for multidisciplinary data. The resulting model that was generated from these requirements is general enough to hold all the different types of data that might arise in a computational aerosciences simulation.

The generalized model is a hierarchy of relationships in which each class "is-a-part-of" its parent class. The relationships amongst the data classes depicted in figure 3.1 are not inheritance relationships. Rather, the hierarchy is a composition of classes that define the necessary components for modeling computational aeroscience simulations. The following paragraphs describe the hierarchical model in order to understand the structure and relationships of the data objects.

The DOMAIN class is the top-level of the generalized model. An instance of the DOMAIN class represents one of the physical disciplines (*i.e.* fluid dynamics, structural dynamics) modeled in a computational aerosciences simulation.

[20]This data model would be considered a data schema if using strict database terminology.

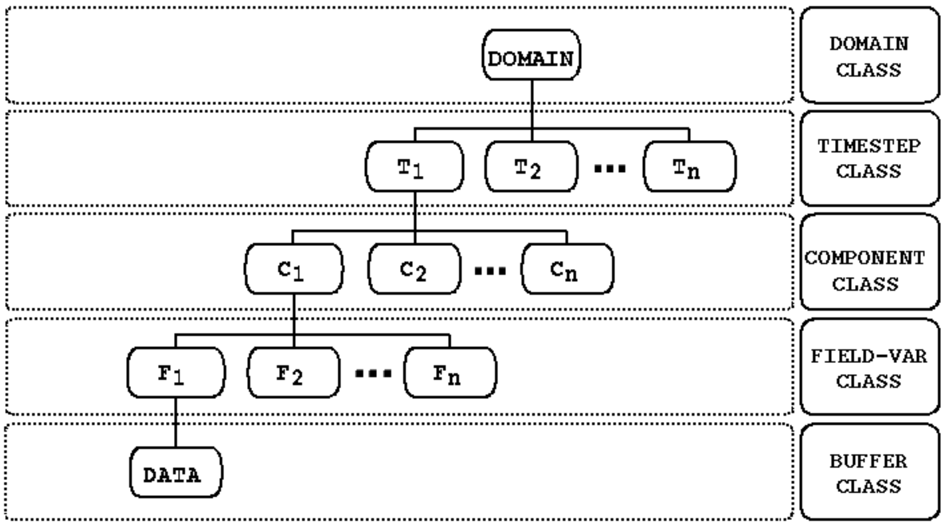


Figure 3.1: The generalized model for time-dependent CAS data. The class composition hierarchy is depicted on the right side of the figure. Each oval on the left side of the figure is an instance of its respective class. Many instances of a class are typically contained within its parent class (*i.e.* several component instances are contained within a single timestep instance.) Additional classes exist that further modify the data hierarchy, however, they are not shown to avoid clutter in the diagram. Only a single branch of the dense hierarchical tree is shown.

The DOMAIN class includes attributes about the domain such as its name, the number of timesteps it contains, and additional information about the domain that contributes to the description of the data.

Each DOMAIN instance contains a sequence, or an ordered group, of instances from the TIMESTEP class. This sequence represents the sequence of timesteps in the time-dependent simulation[21]. Each instance of the TIMESTEP class represents all the data from a single timestep of its domain (represented by its parent class, DOMAIN).

Each instance of the TIMESTEP class contains a collection, or an unordered group, of instances from the COMPONENT class. Members of the COMPONENT class represent the components, or grids, modeled in the simulation. The number of COMPONENT instances can vary from one to several dozen, with each instance representing a unique aspect of the simulated object. This may include physical geometry components such as the wing, fuselage, or flow field phenomena such as a shock.

Contained in each COMPONENT instance are the data fields and dimensions (or topology) that makes up a single grid of a single timestep. The data

[21]If the simulation is time-independent (steady-state), the number of instances of the TIMESTEP class in the sequence is one.

fields include the modeled geometry (position) and the computed data values (*i.e.* momentum, density, energy, deformation, stress). This information is modeled by the COMPONENT class, which contains a collection of instances from the FIELD-VARIABLE class to represent the data variables, as well as an instance of the DIMS (dimensions) class or TOPO (topology) class to store information about the grid connectivity. A COMPONENT with a DIMS represents a structured grid whereas a COMPONENT with a TOPO represents an unstructured grid. Each field in a COMPONENT instance is defined by the same DIMS or TOPO.

Each instance of the FIELD-VARIABLE class contains information about a single field variable in the simulation. Each field variable instance represents only the data computed on a single component at a specific timestep and in a specific domain. For example, a field variable instance can be the position field of the wing at timestep zero in the structures domain. This field variable contains physical locations of the grid points, or the pressure field, which contains pressure values at each grid point. To accommodate grid complexities, field data currently can be of type *node, edge, face* or *cell.* This list allows for data values to be defined not only at the nodes of a computational grid but on edges, faces and cells that are also defined by the grids. For example, the surface normal used for 'flat-shaded' lighting can be defined on a face element of a grid. The default type is *node* for most FIELD-VARIABLE instances. The FIELD-VARIABLE class contains an instance of the BUFFER class, which holds the actual data values. An instance of the BUFFER class contains a storage pointer to the data as well as information about the type of data (float, integer) and the number of items per element. A pointer to the data is used for implementation purposes, so that the system may take advantage of memory-mapping techniques when accessing the data.

For CAS data, each physical domain computed in the simulation is modeled as a specialization of the generalized model. The collection of these domain-specific representations is contained in an object called the MDV-OBJECT. Currently, MDV-OBJECT contains DOMAIN objects representing fluid dynamics (FLUID-DOMAIN) and structural dynamics (STRUCT-DOMAIN) (shown in figure 3.2). MDV-OBJECT represents the top-level, application object of the data model hierarchy. It represents a simulation (*i.e.* HSCT) and all of the data associated with that simulation. Additionally, this object contains attributes about the simulation, initial conditions such as the flight speed of the aircraft (the Mach number), the angle of attack and the given conditions of the airstream (*i.e.* temperature, density).

This section presented the class hierarchy modeling the data from multidisciplinary computational aerosciences. To provide a comprehensive model of the data, attributes, which contain syntactic and semantic information about

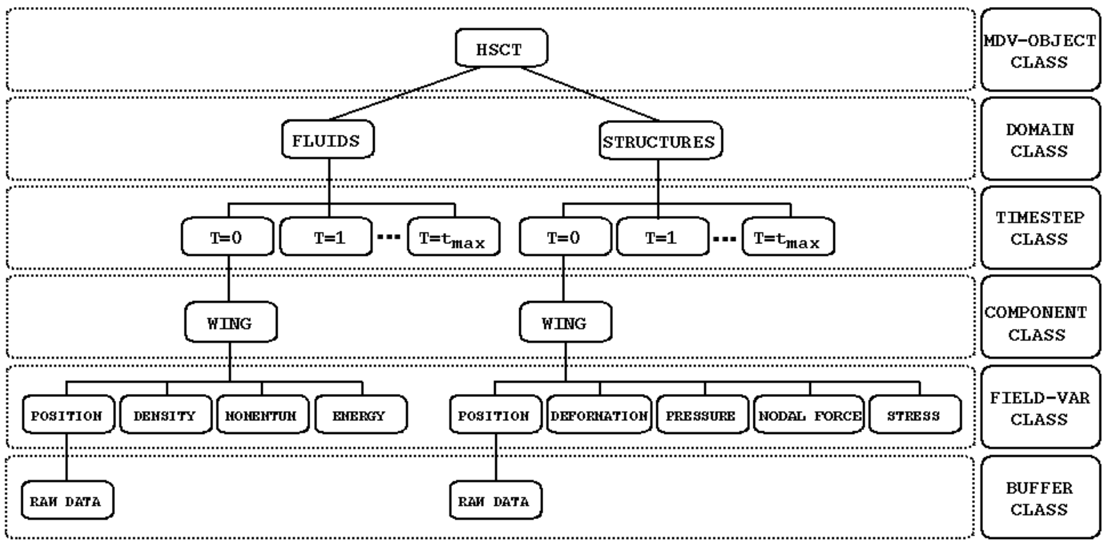


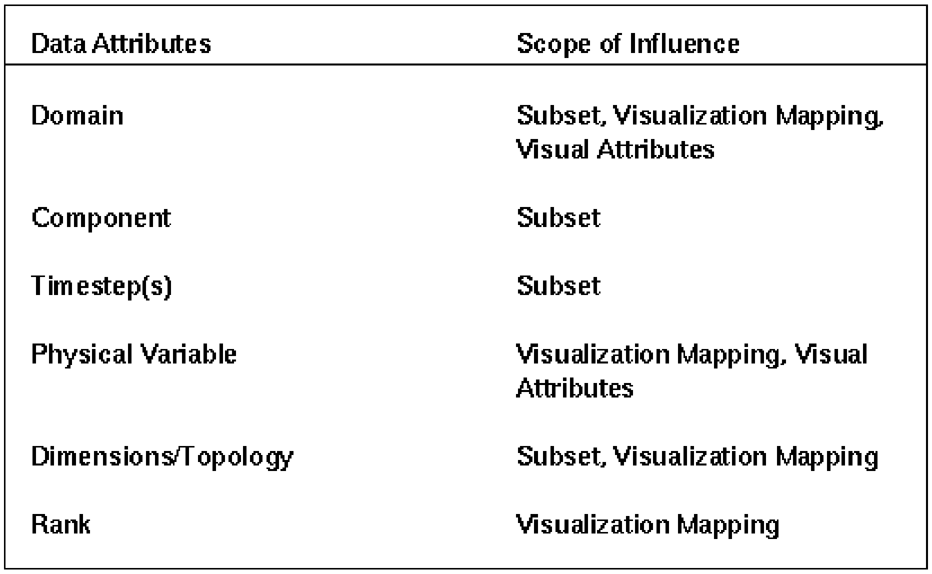
Figure 3.2: A multidisciplinary (fluid dynamics and structural dynamics) model representing the HSCT. Again, the tree has been "pruned" to avoid clutter in the figure.

the data, are necessary. Attributes also guide the visualization assistant with the design of visualizations. The attributes essential to the design process are described in the following section.

3.1.2.3 Data Attributes

The object-oriented data model, with the help of data attributes, captures syntactic and semantic information about data. The following is a description of the data attributes incorporated into the data model. The data attributes influence the design of a visualization in three ways; that is, the "scope of influence" of the attributes can affect 1) the *subset* of the data to be selected for visualizing, 2) the type of *visualization mapping* applied to the data, and 3) the *visual attributes* associated with the rendering of the visualization (*i.e.* the colormap, rendering style). A summary of the data attributes used in designing a visualization is presented in figure 3.3. These data attributes do not fully describe all aspects of the data, but rather provide a base for the prototype visualization system.

Domain: The DOMAIN attribute specifies the physical domain of the data. For this application, the domains are fluid dynamics and structural dynamics. Future physical domains might include thermodynamics or propulsion. The domain may impact which type of visualization mapping is selected based



Data Attributes	Scope of Influence
Domain	Subset, Visualization Mapping, Visual Attributes
Component	Subset
Timestep(s)	Subset
Physical Variable	Visualization Mapping, Visual Attributes
Dimensions/Topology	Subset, Visualization Mapping
Rank	Visualization Mapping

Figure 3.3: Data attributes and their scope of influence in designing a visualization. *Subset* refers to the subset of data the scientist is interested in viewing; *visualization mapping* refers to the type of mapping that is applied to the data subset; and *visual attributes* refers to the attributes associated with the rendering of the visualization.

on commonly used, or standard, visualization mappings in the individual domains. The domain may also affect the visual attributes used in representing the data. For example, a spectral colormap may be used when viewing fluid dynamics data while a "thermal" colormap (with reds and blues representing hot and cold, respectively) may be used for thermodynamics data. Finally, when querying the visualization system, the scientist may chose to only view data from his/her domain. The system selects data from the specified domain.

Component: The COMPONENT attribute semantically describes the one or more computational grids that model the simulated object. The individual grids are used to model either the physical components of the simulated object, such as a wing, fuselage or tail, or the grids are used to capture phenomena in the computed fields, such as a shock in a fluid flow field. The component attribute determines the subset of data to be visualized. The scientist may wish to select one, some or all of the components in his/her query.

Timestep: The TIMESTEP attribute specifies the timestep that the data object represents. In multidisciplinary simulations, the time variable is another independent variable that must be modeled, representing the actual value of time and all of the data associated with it. The timestep attribute determines the subset of data to be visualized. The scientist can choose to view all the timesteps or a subset of them. The timestep attribute may also influence which type of visualization mapping to select for various field variables. For example, if multiple timesteps are selected and the field variable is velocity, the system may suggest the use of a temporal mapping such as the streakline or pathline mapping.

Physical variable: The PHYSICAL VARIABLE attribute semantically describes the field variable, such as pressure, momentum, or deformation. Different physical variables may affect the choice of visualization mappings. For example, the structures variable nodal-force may specify to use a discrete visualization mapping instead of a continuous, since nodal-force is a discrete rather than continuous field variable. The physical variable attribute may also affect the choice of visual attributes. A colortable for viewing temperature data will be best interpreted using reds and blues to represent hot and cold.

Dimensions/Topology: The DIMENSIONS/TOPOLOGY of a dataset represent the size and connectivity of a component grid. Dimensions are relevant when the grid is structured in nature. For unstructured grids, dimensions are replaced by the topology of the grid. Information in the topology include the number of node points and the connectivity of those node points to generate a grid, and includes the number of cells generated as a result of connecting the node points. Both dimensions and topology specify a subset of data to be viewed. The dimensions or topology of the data subset help to select either a local or global visualization mapping. For example, if the dimensions of the grid of data are two-dimensional, the visualization mapping selected should be a local mapping, such as a colormap on the two-dimensional surface depicted by the grid subset. If the subset is three-dimensional, global mappings such as volume rendering or topology may be applied.

Rank: The RANK attribute represents the rank of the data fields computed on any type of grid. Data can be classified by rank of scalar, vector or tensor, with rank zero being scalar, one being vector and two being tensor. The rank of the data will specify what type of visualization mapping to select. For example, scalar data can be represented by the many available scalar mappings such as colormapping and contouring. Vector data can be represented using a vector mapping such as vector plots.

These data attributes are several of the important attributes necessary for designing a visualization. They impact the subset of data to be viewed, the specific visualization mapping chosen and the visual attributes used for rendering. This list is not all-encompassing; however, it is the basic set of attributes implemented in the MDV system. Additional data attributes are contained within the data, such as the data type, the dynamic range, the grid type, the coordinate system, and the data point granularity. However, these attributes do not contribute to the selection of the visualization mapping or its visual attributes.

Data Type: Data classified by data type can be put into one of three categories: nominal data, which have no order, ordinal data, which are ordered by rank only, and quantitative data, which are numerical [6]. The data from multidisciplinary simulations are almost entirely quantitative in nature. Mappings for nominal and ordinal data, such as scatter plots, are not studied in this thesis.

Dynamic Range: The dynamic range of the data specifies the range of values in a given data object or collection of data objects. The range of data is computed based on the subset of timesteps and components specified in the scientist's query. The dynamic range attribute is necessary for the scaling of visual attributes such as a colormap.

Grid Type: Information about the grid type is important when rendering a visualization. Grids can be structured, unstructured, or a hybrid. Grids can also change their shape as the simulation progresses over time.

Coordinate System: In computational aerosciences, the Cartesian coordinate system is almost always used to describe the geometry of the simulated object and its surrounding volume. In computational fluid dynamics, grids are also defined in a "computational space", which is based on the implicit grid connectivity associated with structured grids. Computational space is a logical, regularly gridded space that makes performing visualization mappings easier and faster. In computational structural dynamics, only Cartesian space coordinates are used due to the unstructured nature of the grids.

Data Point Granularity: The information in this attribute specifies whether a data point is of type *node, edge, face* or *cell*. This attribute is necessary to represent data values that are not always computed at the node points of a grid. For example, surface normal data values make sense on a face of a computational grid. In the structures domain, data values may be computed on a two-node element (beam), which is represented as an edge.

Most of the attribute information is input by the scientist when he/she first registers the dataset with the system. This information is then embedded into the data model together with the data that it modifies. Additional attribute information is added to the data model, based on the domain knowledge input into the system by developers.

3.2 The User Model

This section describes the user model and its role in the visualization assistant. The user and machine models are skeletal because they are not the main concentration of this project. Nonetheless, they are integral components of the framework and their description and functionality are discussed. The following sections provide a definition of the user model, a description of how the user model is implemented in the MDV system, and user attributes that it may contain that influence the design of a visualization.

3.2.1 Definition and Purpose

The user model serves to describe the user and his/her characteristics. The information in the user model may include many different user attributes including his/her background, preferences, level of expertise, or physiological characteristics. User background might include information such as the subjective meaning of certain color tables or shades as related to quantitative information. Preferences may include a favorite color table or interaction style. Level of expertise may specify how much experience the user has in visualization. User expertise may affect the complexity of the visualization mappings presented. Physiological characteristics may include a color deficiency or difficulties with fine motor coordination (*i.e.* working with a mouse).

3.2.2 Implementation

The user model is the knowledge source in MDV that contains explicit assumptions about the user that may be relevant to the behavior of the system. The user model is represented using a stereotype hierarchy [38]. A stereotype hierarchy uses stereotypes to describe a general class of users. Stereotypes define facts and assumptions that pertain to the user group's background, preferences and data analysis goals. Models are typically crafted for each application domain, usually by the explicit coding of domain-related goals, plans or knowledge that system users are expected to have. For this application, the hierarchy includes two main user stereotypes: fluid dynamicists and structural dynamicists. Each class of users has a specific academic background which has, in turn, led to the use of certain types of physical models, software, and computing environments.

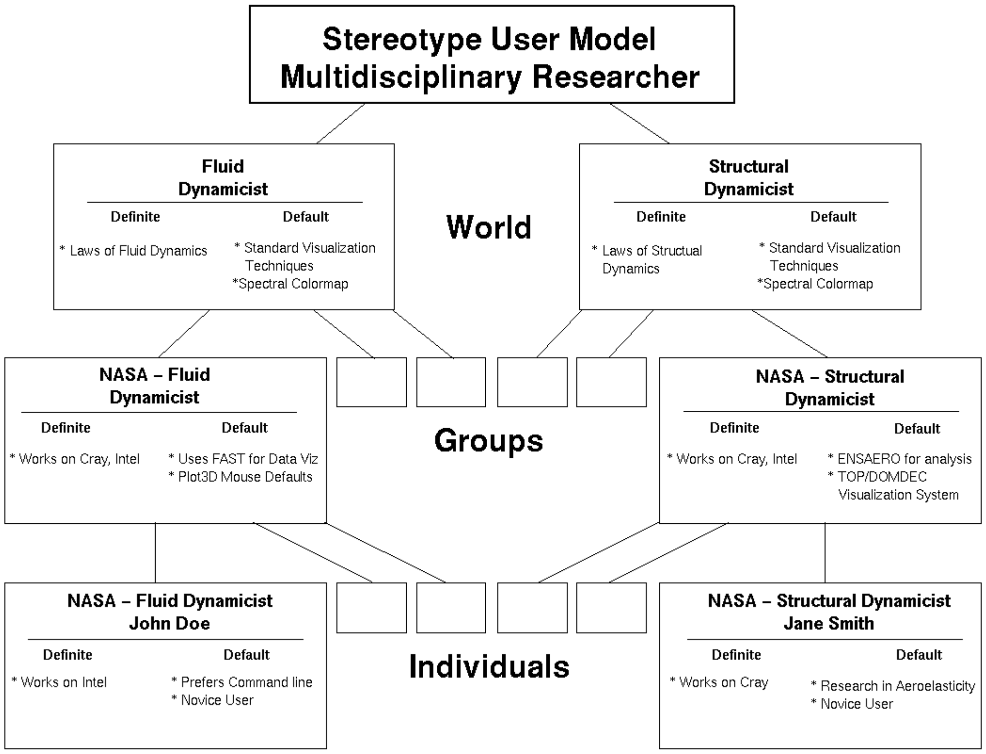
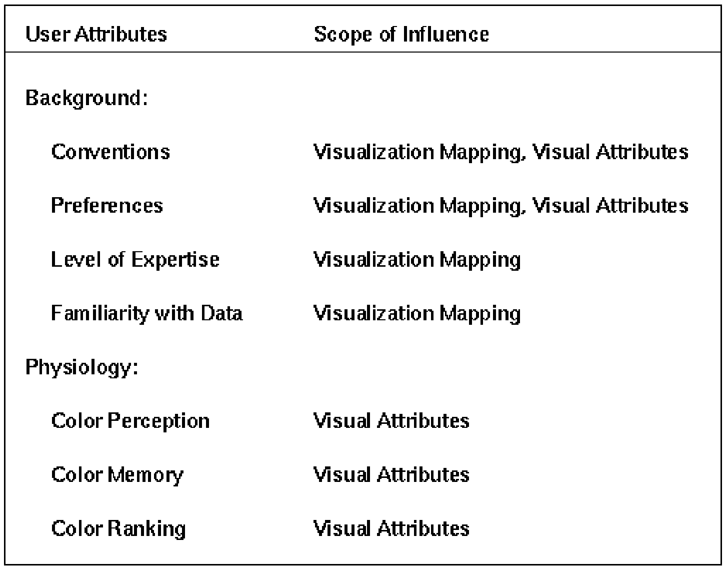


Figure 3.4: The stereotype hierarchy user model.

These facts and assumptions can be of two types, either "definite" or "default". Definite facts and assumptions always apply to the given class of users. For example, fluid dynamicists always use mathematical relationships such as the Navier-Stokes equations to describe the relationships between data variables. Structural dynamicists use Navier's equilibrium equations that describe the deformation and stress of a body when a force or pressure is applied. Definite facts cannot be overridden, but information can be added to make them more precise.

Default facts and assumptions can be overridden as the stereotype hierarchy defines smaller classes of users. For example, a default assumption might be that all fluid dynamicists use structured grids for their computations. However, default assumptions can be overridden if a certain group of fluid dynamicists uses unstructured grids in their work. The stereotype hierarchy can be further decomposed into smaller user classes and eventually to individuals, who are represented as leaf nodes in this hierarchy. The user model is represented in figure 3.4.

User characteristics can be determined by using explicit user modeling, implicit user modeling, or empirical user modeling [24]. Explicit user modeling requires the user to answer direct questions. Implicit user modeling involves



User Attributes	Scope of Influence
Background:	
Conventions	Visualization Mapping, Visual Attributes
Preferences	Visualization Mapping, Visual Attributes
Level of Expertise	Visualization Mapping
Familiarity with Data	Visualization Mapping
Physiology:	
Color Perception	Visual Attributes
Color Memory	Visual Attributes
Color Ranking	Visual Attributes

Figure 3.5: User attributes and their scope of influence in designing a visualization. *Visualization mapping* refers to the type of mapping that is applied to the data subset; and *visual attributes* refers to the attributes associated with the rendering of the visualization.

the system observing the usage characteristics of the user and making inferences. Empirical user modeling involves having the user solve special tasks while being observed by either the system or its developers. These tasks could include tests for color deficiencies or for difficulties with fine motor coordination (when using a mouse).

3.2.2.1 User Attributes

User attributes that affect the user's performance and interpretation of data fall into different categories, such as background and physiology. These user attributes affect the generation of a visualization in two ways, 1) the type of *visualization mapping* to be selected, and 2) the *visual attributes* associated with its rendering (i.e. the colormap, rendering style). A summary of the described attributes together with their scope of influence is shown in figure 3.5.

The specific background of a scientist can affect how he/she interprets a visualization. The scientist's domain of study maintains its own conventions, color schemes, and methods for data analysis. The scientist brings in his/her own personal preferences and experience level. This information is necessary to select visualization mapping matched to the capabilities and preferences of the scientist.

Conventions: Conventions are methods for doing things a specific way given a specific scientific domain or working group. For example, fluid dynamicists typically use contour lines that do not have numerical values embedded in them. Instead, they often color code their contour lines to distinguish data values. This drawing style is a common convention that has propagated during the evolution of visualization systems for computational fluids. This attribute can affect the type of visualization mapping selected and the visual attributes used for rendering.

Preferences: Preferences are individual, representing the scientist's favorite way of doing things. For example, a scientist might prefer using a certain colormap for aesthetic reasons. By incorporating information about the scientist's preferences, the resulting visualization is more pleasing. This attribute can affect the type of visualization mapping selected and the visual attributes used for rendering.

Level of Expertise: Scientists who are interested in using visualization as an analysis tool come with many different levels of experience. For example, some scientists are extremely knowledgeable about visualization and enjoy creating several different representations of their data, some of which are very complex. Others are interested in only generating simple images. The complexity of a visualization mapping should correspond to the experience level of the scientist. Many simple visualization mappings are easy to interpret, based on intuitive physical models. However, the some mappings used to visualize vector or tensor fields (i.e., vector field topology [27] or tensor field visualization [17]), are complex and require training on how to interpret the resulting visualizations. The use of these mappings is very valuable for the scientist who is willing to invest the time to gain the additional information they can offer. More complicated visualizations may be suggested to the scientist as they gain experience with the data.

Familiarity with Data: Another dimension of the user model is the scientist's familiarity with the data he/she is analyzing. If the data is completely new to the scientist, he/she is probably interested in viewing simple representations of the data, preferably in a highly interactive and exploratory mode. If the scientist is familiar with the data, he/she may want to see a more sophisticated visualization to bring out features that have not been seen. These sophisticated mappings might require more computational resources to generate. A good example is volume rendering, which can consume several minutes of compute time. A "committal factor" on the part of the scientist is associated with his/her familiarity with the data. If the scientist is closely analyzing a specific dataset and is committed to understanding it very well, he/she is more willing to wait to view such a representation. On the other hand, if the scientist is exploring his/her dataset, he/she will be more interested in rapidly generated visualizations.

The user's physiology may influence the color schemes and manipulation methods used for visualization. The following is a description of some of the physiological attributes of a user model, based on work by Domik and Gutkauf [21, 29]. These aspects of the user model have not been implemented in MDV. However, they are the basis for future work in generating a more complete user model for the MDV system. The following three attributes influence the selection of visual attributes in the design of a visualization.

Color Perception: The perception of color is a very subjective, cognitive ability that differs for every user. Color perception can be affected by color deficiencies, gender, ethnological information, output medium and the user's environment (*i.e.* lighting in the room). For example, color deficiencies, which are caused by missing color receptors in the retina of the user's eye, can affect the way users perceive different colors. The three types of color blindness are protanopia (missing long receptor), deuteranopia (medium receptor) and tritanopia (short receptor). Protanopia and deuteranopia are most common and occur in approximately eight percent of the male population and one percent of the female population. The standardized Farnsworth-Munsell color test can be implemented on a CRT device to test for this difficulty. Color deficiencies cause "confusion lines" in perceptual color space [46]. To manage such a deficiency, specialists suggest creating color tables orthogonal to these confusion lines.

Color memory: When interpreting data via visualization, the scientist must often recognize color and remember it from image to image. The color memory of a user is their ability to recall these colors and remember distinctions between colors for semi-quantitative analysis. Humans are able to distinguish as many as 10 million colors [36], provided we have no deficiencies. However, this is only possible when colors are directly compared with one another. Without direct comparison, the number of colors we can process and individually identify is much smaller. Color memory tests can be administered to identify which colors a user remembers most accurately. This information can then be applied in selecting colors and color scales to be used in future visualizations.

Color ranking: Ranking colors involves associating their perceived color with either ordinal or quantitative values. A common example is the use of reds, yellows and oranges to represent hot values in a temperature scale, while blues represent cool values. Since human perception has no inherent ranking of colors, most of these associations are learned from either natural phenomena or from standard mappings in one's own discipline. If these rankings can be identified, then appropriate scales can be created for the individual to help them to analyze data.

The simplified user model for the MDV project incorporates only a subset of the user attributes described above. User information was collected in an explicit user modeling fashion, by directly interviewing scientists. This information was incorporated into the user model and is accessed according to the username of the individual currently working on the MDV system. The attributes incorporated in MDV include the background characteristics of the scientists performing the fluid/structure simulation. These attributes include information about conventions in the fluid dynamics and structural dynamics domains. Preferences of the individual scientists include their choices for colormaps. "Level of Expertise" and "Familiarity with Data" are not yet incorporated, as the set of visualization mappings currently available in MDV is still very limited. The scientists are assumed to be novice users with little familiarity with the data. The user model does not take into consideration the physiological characteristics of the users, nor does it attempt to model any additional characteristics at great length. The purpose of the skeletal model is to understand how a user model integrates into the overall framework. Future work will involve expanding this aspect of the framework, incorporating additional information about scientists that might be relevant to the behavior of the system.

3.3 The Machine Model

This section describes the machine model and its role in the visualization assistant. Similar to previous sections, this section defines and describes the machine model, its implementation in the MDV system and the machine attributes contained in the model that influence the design of a visualization.

3.3.1 Definition and Purpose

The machine model details the capabilities of the computing environment available for scientific use, including both software and hardware resources. The scientist should not be expected to know the characteristics of his/her hardware and software environment. This information is important, however, as it impacts what type of visualization will be appropriate for a given configuration. Information will be contained in the machine model, specifying the type of system, monitor resolution, graphics capabilities, color facilities, CPU, memory, and available graphics libraries. This model should describe a wide spectrum of machines, from laser printers (for publication) to workstations to virtual reality systems. The incorporation of the machine model makes the system very portable, allowing the scientist to analyze his/her data using different environments.

3.3.2 Implementation

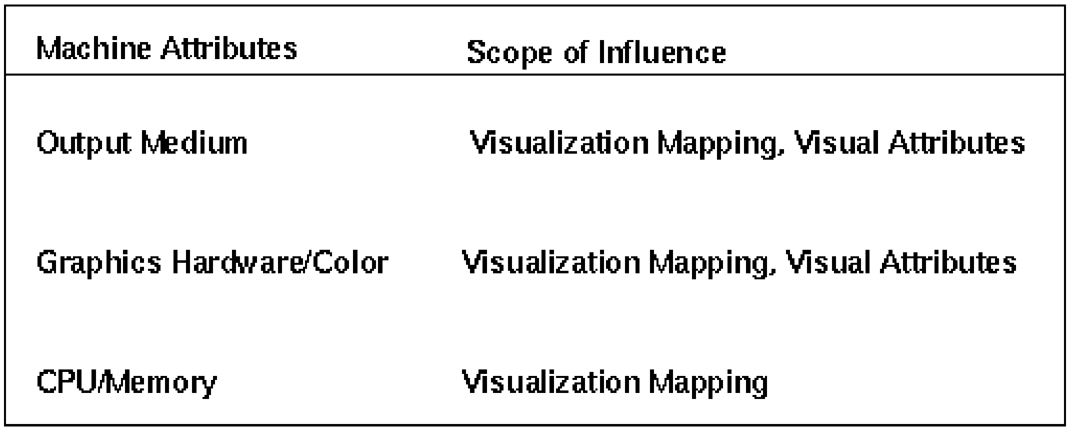
MDV incorporates a simplified machine model, describing the current hardware and software used by the scientist. The information in the machine model is modeled using the object-oriented paradigm with each hardware device modeled as a specific object. Currently, the MDV machine model describes only three instances of hardware devices: a workstation, a color printer and a black and white printer. The workstation object represents a Silicon Graphics (SGI) Indigo R4000 workstation, and the printer objects represent a Tektronix color printer and a standard laser printer.

3.3.2.1 Machine Attributes

The properties of the hardware and software environment of the scientist can affect the selection and presentation of a visual representation in many ways. The following attributes, taken from [39], detail some of the considerations that must be taken into account. These machine attributes influence the design of a visualization in two ways, 1) the type of *visualization mapping* to be selected, and 2) the *visual attributes* associated with its rendering (i.e. the colormap, rendering style). A summary of these attributes and their scope of influence is shown in figure 3.6.

Output Medium: The output medium is the device on which the suggested visualization will be displayed. The average user takes advantage of several output media such as the computer monitor (CRT), plain paper printouts, and transparency printouts. The selected output medium affects the visualization mapping and visual attributes.

Computer monitors employ additive color generating mechanisms, beginning with a black screen and combining components of red, green and blue to generate a specified color. Printers, on the other hand, generate colors in a subtractive manner, beginning with a white or clear page and subtracting cyan, magenta and yellow (black is also used) to produce the designated color. The various color-generating techniques can lead to differences in the final



Machine Attributes	Scope of Influence
Output Medium	Visualization Mapping, Visual Attributes
Graphics Hardware/Color	Visualization Mapping, Visual Attributes
CPU/Memory	Visualization Mapping

Figure 3.6: Machine attributes and their scope of influence in designing a visualization. *Visualization mapping* refers to the type of mapping that is applied to the data subset; and *visual attributes* refers to the attributes associated with the rendering of the visualization.

output image. For example, combining red and green yields yellow on a CRT, but on a color printer can produce a muddy brown color. Colors that are clear on the screen are not faithfully reproduced on many color output media. If the output medium is a black and white printer, the visualization mapping selected must not rely on the use of color for interpretation. Good mappings to use in this case are contour lines and vectors, which display the detail of the data expressively on the output medium.

For an expressive visualization, the background color of the image is an important consideration and dependent on the output medium. For example, if transparencies are generated for presentation purposes, areas of interest drawn in light colors, such as white, may be difficult to see when projected.

The resolution of the resulting image (i.e. pixels on the CRT, dots per lineal inch on a printer) is important in creating a high quality output that does not lose any information detail. If the suggested visualization is rendered to lower quality output medium, such as video, details will become less distinct and colors will become fuzzy. If the visualization is rendered to higher quality output medium, such as a 300 dot per inch laser printer, details that are represented by small dots may become too small to distinguish. The aspect ratio of an image may be changed when it is reproduced on various output mediums. This change must be accounted for to ensure that scales are maintained.

Graphics Hardware/Color: CRTs are available with full 24-bit color that allows for "true" representation of the RGB color space. Lower quality CRTs are also prevalent that use 8 bits to depict "pseudo-color". These CRTs limit the range of colors that can be represented and may influence the color scale selected or the type of visualization to be displayed. Sophisticated visualization mappings, such as those generated using texture mapping techniques [13], may require sophisticated graphics hardware. If this hardware is not available, the time to produce a visualization may become prohibitive.

CPU/Memory: Visualization mappings that are compute and memory intensive may require a certain amount of CPU power or memory. If this minimum is not available, generating the visualization may become infeasible.

The machine model currently describes three devices, a workstation, a color printer, and a black and white printer as output mediums. Information about their processing speed, memory, etc. is not included in the machine model. The machine model aspect of the visualization framework is skeletal and future work will involve expanding the machine model to provide a more comprehensive description the scientist's working environment.

3.4 The Knowledge Base

The knowledge base is a collection of assumptions about data, user, and machine attributes that are relevant to the design of visualizations. These assumptions are based on principles from perception and graphic design, and information from the individual scientific domains. The intent of these assumptions is to provide the necessary information for making appropriate choices when designing a visualization. For example, an assumption about the data object FLUIDS/TIME=0/WING/DENSITY (shown as a leaf node in figure 3.2) might be that a commonly used color table for viewing density is the spectral color table. This assumption was taken from fluid dynamics domain knowledge. An assumption about the machine object COLOR PRINTER would be preferable background and foreground colors to use for the visualization. This assumption was taken from knowledge in graphic design.

The knowledge base of assumptions is not a distinct entity in the visualization framework. Rather, the assumptions are embedded into the individual models based on the information source they describe. The assumptions are considered as additional semantic information and are implemented in the information models as attributes. Using the examples in the previous paragraph, the data object FLUIDS/TIME=0/WING/DENSITY contains an attribute COLOR TABLE that contains the value SPECTRAL to note that an appropriate choice for color tables is the spectral color table.

The design decision was made to integrate the information models and assumptions rather than explicitly stating the assumptions in a separate knowledge base. For the assumptions used in this project, this design decision proved to be adequate. In the previous examples, the assumptions included in the knowledge base are relevant to specified objects in the individual models. By storing the assumptions with the information they modify, a clearer semantic description of the information and its attributes is achieved. The colocation of the information models and assumptions reduces the complexity of storing the assumptions in a different form in a separate knowledge base. In addition, assumptions are easy to add and modify, since they are easily added to the models in the form of attributes.

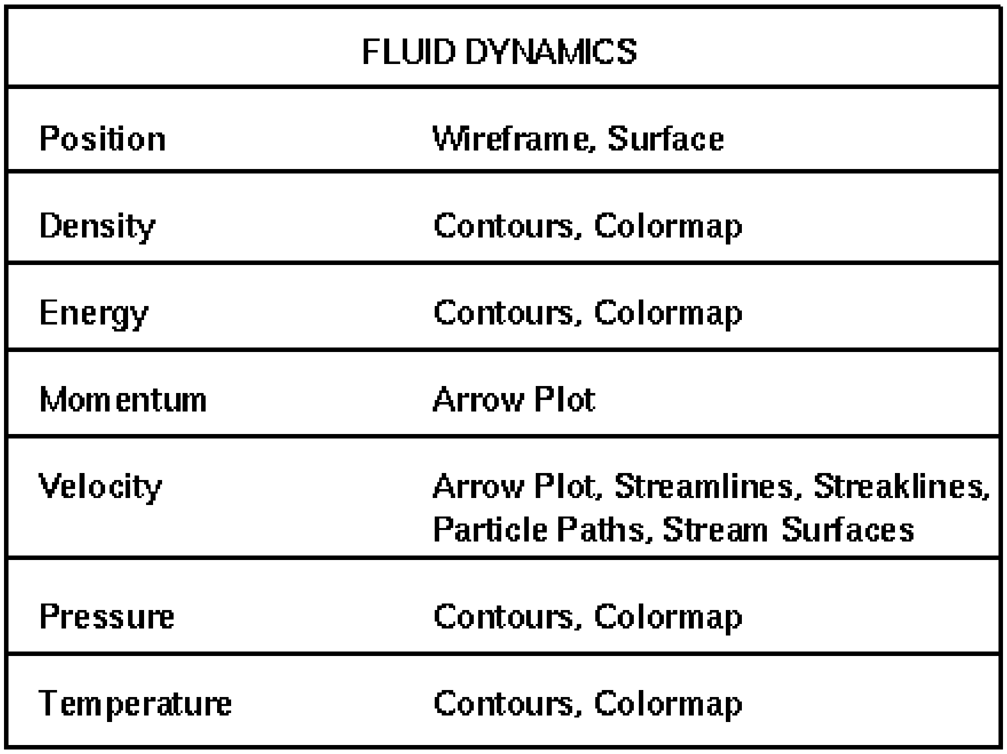
The decision to integrate the assumptions with the models has potential problems with regard to the management of the assumptions and performance. Because assumptions are implemented as attributes, they are intended to represent an assumption about a single information source, such as the data model. Assumptions that may apply to a combination of multiple sources, such as the data model and user model, are not represented as easily. For example, an individual fluid dynamicist prefers to view the density field variable using a greyscale color table. This assumption applies to both the data model and user model. Although this assumption can be implemented in one model in terms of information contained in the other model, the representation is not as simple. The above assumption would be implemented in the user model representing the individual in terms of information contained in the data model. The assumption in the individual's user model would state that if the field variable to be viewed is density, select the greyscale color table. In addition, because assumptions are spread throughout the models rather than consolidated into a single model, the process of designing a visualization may be more time consuming.

The assumptions contained in the visualization assistant are based on principles from perception and graphic design, and domain knowledge. The following sections describe these information sources.

3.4.1 Perception and Graphic Design

The knowledge base contains assumptions that contribute to the design of effective and expressive visualizations. These assumptions are based on principles from graphic design and visual perception and the advice and knowledge of scientists in the different scientific domains.

Principles from information display have been documented by such researchers as Bertin [6] and Tufte [66, 67]. Perceptual rules and graphic design guidelines are also readily available [16, 19, 37, 39, 44]. Information has also been collected by researchers specifically for scientific visualization, including



FLUID DYNAMICS	
Position	Wireframe, Surface
Density	Contours, Colormap
Energy	Contours, Colormap
Momentum	Arrow Plot
Velocity	Arrow Plot, Streamlines, Streaklines, Particle Paths, Stream Surfaces
Pressure	Contours, Colormap
Temperature	Contours, Colormap

Figure 3.7: Standard visualization mappings for fluid dynamics data.

Senay and Ignatius [60] and Wehrend and Lewis [70]. A sampling of these principles is presented in Appendix A.

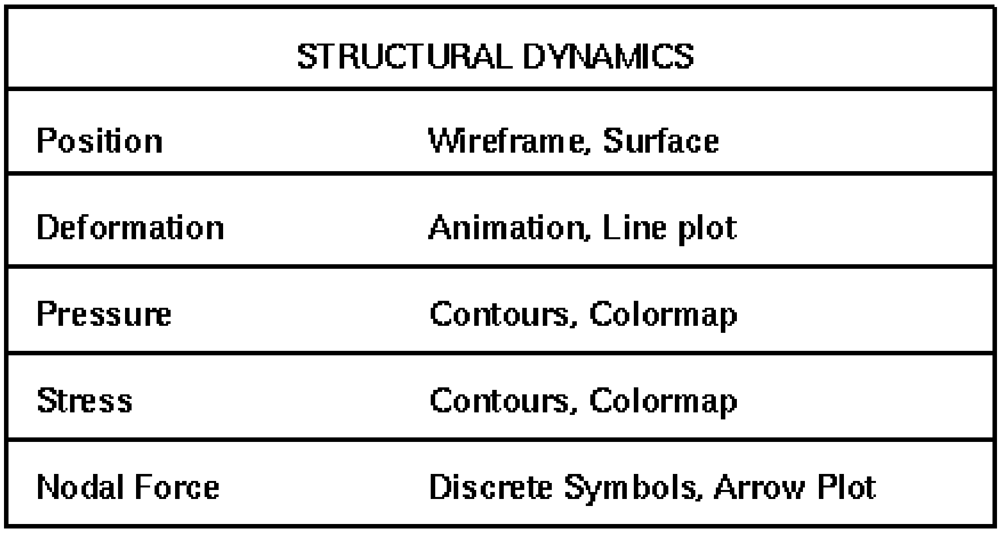
3.4.2 Domain Knowledge

The knowledge base also includes domain dependent assumptions that influence the design of visualizations. This information includes assumptions on how to effectively visualize the different data fields of each dataset. Commonly used, or standard, visualization mappings for each data variable are attached to that field within the data model. These standard mappings were obtained by working with scientists and understanding what types of mappings are appropriate for different data fields. Examples of "standard visualization mappings" are shown in figure 3.7 and figure 3.8.

Domain knowledge may also provide information as to an appropriate color table for visualizing a certain data variable in a certain domain. This information and addition domain dependent information is listed in Appendix B.

3.5 Designing a Visualization

Designing a visualization is an iterative process that uses the information and assumptions stored in the data, user, and machine models and the



STRUCTURAL DYNAMICS	
Position	Wireframe, Surface
Deformation	Animation, Line plot
Pressure	Contours, Colormap
Stress	Contours, Colormap
Nodal Force	Discrete Symbols, Arrow Plot

Figure 3.8: Standard visualization mappings for structural dynamics data.

knowledge base. To design a visualization, a visualization specification (to be described in Section 3.5.1) must be specified by the scientist. Briefly, this involves designating a subset of the data to be visualized and a visualization task. Based on this specification, the system extracts the requested data and assumptions about the data, user and machine environment, and designs and renders a visualization.

Assumptions from the data model provide initial information on what type of visualization mapping is most appropriate given the data variables of interest. Data model assumptions also suggests visual attributes such as a standard color table for viewing the chosen field data. The selected visualization mapping and visual attributes are modified by the user model where information on user preferences might select one color table over another. This result is then modified by the machine model, where additional changes may be required by the type of output the user has selected. For example, if the scientist is printing to a black and white printer, the system might suggest contour lines colored using greyscale values.

If some constraints are violated, the process returns to the data model where an alternate visualization option is selected. This selection is also modified by user and machine models. This process continues until a satisfactory visualization is found or until the system runs out of suggestions. Once this information is selected, it is applied to the appropriate data and displayed for the scientist. The generation cycle is displayed graphically in figure 3.9.

This method for designing the visualization was selected because of its simplicity and ease-of-implementation. It is not a very sophisticated reasoning process and could be problematic if the knowledge base of assumptions were

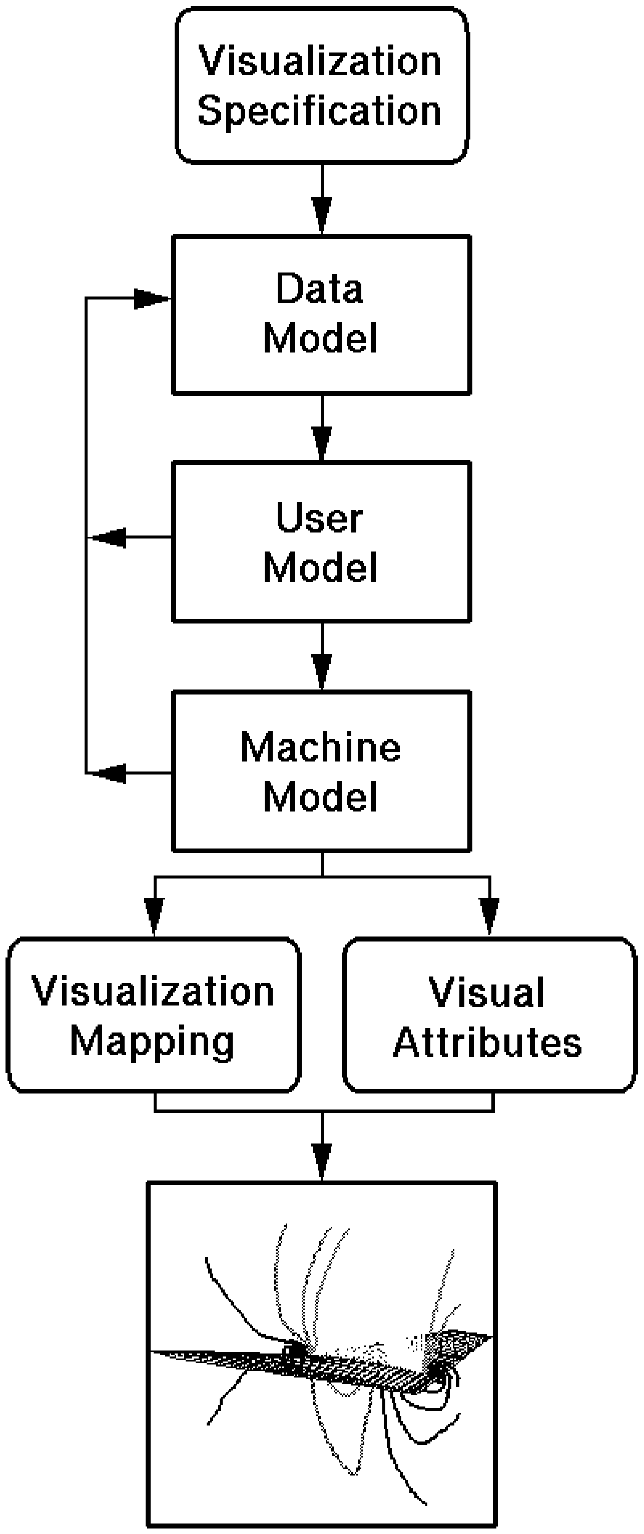


Figure 3.9: The visualization generation cycle.

to grow very large. Currently the knowledge base of information is not very large, so the process of designing the visualization is not time-prohibitive.

One of the factors affecting performance is that the process of designing a visualization cycles through the data model, user model and machine model until a satisfactory visualization is designed. This is clearly not an optimal solution for designing the visualization. Optimizations should be investigated so that this process could be made more efficient. An alternate method to use in the design process would be the incorporation of a knowledge-based system that contained a reasoning engine. This option is discussed further in section 4.1.1.

3.5.1 Visualization Specification

The visualization specification establishes the visualization goal of the scientist. The visualization specification contains two components. The first component is the selection of the data subset to be displayed. The second component is the selection of the task to be applied to the data subset. The visualization specification should be processed quickly and the scientist should be presented with a visualization that satisfies the query. The scientist should not be required to learn a complicated query language to create the visualization specification.

3.5.1.1 Data Subset Selection

The scientist must select the subset or portion of the dataset which they would like to view. The physical domain, timesteps, components, grid subset, and field variables are all part of the data subset specification. The selection of the data subset consists of two parameters, "scope" and "focus of attention", taken from Roth [28]. The first parameter, scope, selects the amount of data one wishes to view. In MDV, selecting the scope involves selecting the physical domain, timesteps, components and grid subsets of interest. This process reduces the amount of data to be visualized to a manageable amount. The second parameter, focus of attention, selects which attributes of the data the user wishes to view. This selection involves choosing the physical variable of interest, such as pressure or stress.

3.5.1.2 Task Specification

Many systems and research projects, such as [3, 14, 39, 70], state the need to describe the task or interpretation aim the scientists is interested in accomplishing. The task or interpretation aim asks the scientist questions such as "What am I trying to accomplish by viewing this data?", "What do I hope to learn from this data?" and "What do I want this data to show me?".

From the research mentioned in the previous paragraph, two levels of task specification have been established. The first is a low level, detailed specification that involves identifying the relationships the scientist is interested in extracting. The second is the high-level, goal-oriented accomplishment that the scientist is interested in performing.

Low-Level Task Specification Low-level task specification involves identifying very basic relationships between data points or regions of data. Wehrend [39], defines user's visualization goals as the information the scientist is expecting to learn and communicate in his/her data. These goals include: identify, locate, distinguish, categorize, cluster, rank, compare, associate, and correlate. Keller and Keller [39] list the following tasks within visualization: comparing, distinguishing, indicating direction, locating, relating, representing values, and revealing objects.

High-Level Task Specification High-level tasks identify common operations performed by scientists during their data analysis sessions. These tasks can best be identified by examining typical usage scenarios performed by the scientist, such as was done in section 2.3.2. It was determined from these scenarios that within the scientific domains visualization is often used to perform high-level tasks such as DEBUGGING, EXPLORATION, ANALYSIS and PRESENTATION.

The DEBUGGING task is an essential part of the development process where simple visualizations (such as wireframes) are presented to help debug areas of code. EXPLORATION involves a trial and error process with the data, and interactivity is important. ANALYSIS is a study of known relationships, which may require metrics or some precise means of measurement (i.e. integrating qualitative and quantitative representations). Finally, the PRESENTATION task requires aesthetic, annotated and intelligible graphics. In addition, the size of the audience is typically larger, so the use of complex visualization mappings should be used carefully.

The MDV system approaches the task specification from the high-level, because feedback from scientists suggested that the low-level task operators were difficult to apply to their data analysis goals. The high-level task operators specified are undoubtedly only a subset of the tasks performed by scientists. These four operators are also very simplified and may not sufficiently describe the task the scientist has in mind. Finally, because the data analysis task of the scientist is difficult to capture, the four tasks selected for implementation may overlap in their functionality. For example, the scientist may not be able to distinguish when he/she is interested in EXPLORATION or when he/she is interested in ANALYSIS. Also, the scientist may progress from EXPLORATION to ANALYSIS using a single visualization and may not be interested in designing another visualization. The issue of determining tasks is a complex one that will involve a great deal of study into the way scientists interact with their data.

3.6 Summary

This chapter presented the visualization framework, describing the necessary components for creating a visualization assistant. The basic framework, consisting of a data model, user model, machine model and knowledge base, can be incorporated into the design of any visualization system regardless of the application domain. Included in the descriptions of the models was the identification of the attributes in the data, user and machine models that affect the design of a visualization. Most of these attributes are general and can be incorporated into the framework as necessary for different application domains. An analysis of the specific application domain will reveal additional attributes that influence the design of visualizations for that domain.

The object-oriented model was selected over other modeling paradigms because its modeling capability is superior for handling hierarchical, multidimensional, scientific data. For example, it is difficult to handle the simple interrelationships found in a two-dimensional array using the relational model [5]. In addition, it is difficult to capture semantic information about the data using the relational database model. The special purpose Fiber Bundle model does not contain mechanisms for handling semantic information. Semantic information is necessary for describing the data attributes that guides the design of a visualization.

The purpose and contents of the knowledge base was also presented, identifying the sources of visualization knowledge to be used for visualization design. The information in the knowledge base, specifically, information from perception and graphic design, could also be applied to different applications since it is based on basic principles. Domain dependent knowledge is also essential and must be gained in order to select appropriate mappings and visual attributes.

Finally, the process of designing a visualization, as performed within the MDV system, was defined. The first step of this process requires the scientist to specify the subset of data to be visualized and the data analysis task. Data analysis tasks were extracted based on the scenarios defined in section 2.3.2. High-level tasks, such as DEBUGGING, EXPLORATION, ANALYSIS, and PRESENTATION, were used to define the goal of the scientist using the visualization tool. The use of high-level, instead of low-level tasks, is in contrast to previous research done in the development of automated presentation and visualization tools. The tasks selected were based on user feedback and analysis of typical operations performed by scientists at the high-level. The tasks selected were not comprehensive and somewhat overlapping in function. This aspect of task selection is understandable, however, as the set of tasks performed by the scientist is representative of a continuum rather than a discrete set.

The visualization design process was useful for the initial prototype. However, it was evident that for a production system this method may not be sufficient for reasons of performance and knowledge management capability. Other mechanisms for managing the necessary knowledge may be necessary, such as the use of a commercial knowledge-based system. The use of knowledge-based systems is considered and a further discussion on implementation is presented in section 4.1.

CHAPTER 4
THE MDV PROTOTYPE

This chapter discusses the implementation of the visualization framework described in Chapter 3. A description of the programming environment selected for the project is presented. The final product, MDV or MultiDisciplinary Visualizer, is then described, detailing the functionality of the system and its user interface.

4.1 Implementation Environment

To implement the MDV prototype, it was necessary to choose an environment that possessed the following characteristics:

1. Support for modeling the visualization framework described in Chapter 3;
2. Capability to handle large, multidisciplinary scientific data;
3. Reasoning mechanisms to manipulate the information in the framework for visualization design;
4. Visualization functionality to render and interact with designed visualizations.

To meet these diverse requirements, two options were considered: the use of a knowledge-based system and the use of a visualization programming environment. These options are presented in the next several sections, followed by a section on the environment eventually selected for the visualization assistant.

4.1.1 Knowledge-Based Systems

Several knowledge-based systems are available commercially or in the public domain, including KEE [1], KL-ONE [11], and CLASSIC [51]. Implementing the visualization assistant using a knowledge-based system would have been a feasible approach. Knowledge-based systems provide mechanisms for modeling and managing data and knowledge as well as reasoning engines to manipulate the knowledge.

For example, the KEE system, developed by Intellicorp, uses a frame-based representation for modeling data and knowledge. The frame-based approach is similar to the object-oriented paradigm, modeling related information in a high-level, abstract manner. Typical frame language constructs represent classes, attributes, and methods. Frames can describe data or they can contain forward and backward chaining rules relevant to the data. Frames containing rules have premises, assertions, actions and control parameters that are stored as slots (attributes) and provide reasoning functionality.

Implementing a visualization assistant, such as MDV, would be feasible using the KEE system, representing data, user, and machine characteristics with frame-based constructs. Visualization assumptions could be implemented using the constructs described for managing rules.

The work performed by Ahmed, et al. on the Intelligent Visualization System (IVS), is an example of a knowledge-based approach to the design of a visualization assistant. In the IVS, the CLASSIC knowledge-based system was used. CLASSIC maintained rules regarding the construction of data-flow networks. These networks were then input into the AVS system for visualization. A separate data model (fiber bundle model) was used to manage the data.

The advantages to using a commercial knowledge-based system are evident. The software framework is already available for the visualization design process. The information specified in Chapter 3 could be modeled using constructs available in KEE. The KEE system is well suited for handling the visualization knowledge, or assumptions, and using it to design visualizations. KEE is also suited for large amounts of knowledge, using a relational database as its back end. As the assistant expands its knowledge base, KEE would be able to manage the additional information. Finally, since KEE has been developed and optimized to handle knowledge in a frame-based format, KEE's reasoning mechanisms are likely to be efficient.

However, some disadvantages are also evident. Although KEE supports large amounts of frame-based information, it is unclear if KEE would be able to manage the large datasets associated with scientific data. Knowledge based systems are typically suited for a large number of small records, but not necessarily for large chunks of scientific data. Using a knowledge-base system such as KEE may have required the incorporation of additional software to store and manage the data. In addition, visualization software would also have to be integrated with the knowledge-based system for presenting the designed visualization to the scientist. The resulting visualization assistant would be a collection of software tools that have to communicate information and data amongst each other. The additional overhead in integrating several systems could jeopardize performance and increase complexity.

A knowledge-based system, such as KEE, directly met two of the four requirements specified at the beginning of this section for the implementation of the visualization assistant: the modeling capability (item 1) and reasoning capability (item 3). The knowledge-based approach did not possess visualization functionality and it was unclear if data management capabilities would be acceptable.

4.1.2 Visualization Environments

An alternative implementation option for MDV was a visualization programming environment. Several visualization environments are available that allow programmers to build visualization modules. These include systems such as FAST, Explorer, and AVS. However, these systems do not have the capability to incorporate the information described in Chapter 3.

A visualization environment must provide the features of a programming environment coupled with data management and visualization primitives. An object-oriented visualization programming environment would keep the system conceptually simple. In addition, this type of environment would tightly integrate the data model and visualization mappings for better performance in handling data and computing and rendering visualizations. Transfer of data and information between the visualization framework, the knowledge-based system and the visualization system would not be necessary.

The SuperGlue [35] environment, designed specifically for creating flexible visualization applications such as the MDV visualization assistant, was selected for implementation. SuperGlue directly met three of the four requirements for the implementation of the visualization assistant: the modeling capability (item 1), the capability to handle large, multidisciplinary data (item 2) and the visualization functionality (item 4). The main feature missing from SuperGlue was the reasoning engine necessary to design the visualizations. However, because of the prototype nature of the project, it was determined that this was an acceptable compromise. The knowledge base associated with the prototype system was small and a simple mechanism for manipulating the assumptions in the information models was sufficient to develop a proof-of-concept system. The main concentration was placed on mapping out the visualization framework and identifying the attributes necessary for designing visualizations.

4.1.3 SuperGlue

MDV has been implemented in SuperGlue, a rapid prototyping programming environment under development at the NASA Ames Research Center. SuperGlue is an object-oriented programming environment based on the language Scheme [8], a dialect of Lisp.

SuperGlue provides an effective environment for the rapid prototyping of code due to its interpreted nature. The SuperGlue class mechanism most closely resembles that of Smalltalk. SuperGlue provides single inheritance and dynamic method-lookup based on the message and the class of receiver. SuperGlue also provides dynamic typing, allowing each variable to hold values of any type. Checking for mismatches occurs at runtime. This dynamic typing capability allows the same piece of code to be used on different types of data, supporting a rich variety of data types. Automatic memory reclamation (garbage collection) frees the programmer from keeping track of the storage used in the program.

SuperGlue provides a large object-oriented class hierarchy, which contains components necessary for generating tools for interactive visualization. The class system is divided into subtrees which implement generic functions for visualization. This hierarchy provides classes for data management, process management, window management and event handling, mathematical functions, visualization mappings, and rendering, among others. Information is easy to incorporate into the SuperGlue hierarchy, making easily extensible as the application's functionality and information base grows.

SuperGlue has the capability to incrementally load compiled foreign functions written in traditional languages such as C and FORTRAN. Computationally intensive operations are typically performed in C. Scheme is used for high level control, management of data objects and for user interface handling.

The performance of the visualization environment is reasonable considering the large amounts of data to be digested and displayed. By allowing C to perform computationally intensive operations (as well as graphics operations), MDV is not hampered a great deal by performance considerations. Optimization of the MDV system would produce a more efficient system, however, this is not a main goal of the project.

SuperGlue uses memory-mapping techniques. As a result, the actual data is only paged into memory when it is queried by the scientist. This provides necessary performance when working with large, time-dependent datasets.

The Tcl and Tk [49] package can also be accessed from SuperGlue for the creation of user interface widgets. Tcl language is a command language. Tk is an X11 toolkit, written in Tcl, that provides the look and feel of Motif widgets. Access to the Tk widgets is provided through Scheme commands within SuperGlue. This accessibility allows for rapid production of user interface widgets for the visualization software.

4.2 The Final Product

The current system is still a prototype by any standard, but it is functional and available for use by the scientists. It provides scientists with the capability to view their data as wireframes, surfaces, contours, colormaps, particle traces, and vector plots. The following section describes the user interface and functionality of the MDV system.

4.2.1 The User Interface

The user interface allows the scientists to specify their data analysis goals at a high level. The system takes the query made by the scientist and generates a visualization from it. The query is easily specified by the scientist, using a widget-based interface.

MDV can be started from the command line. Upon entry, MDV obtains the scientist's ID and machine name using standard UNIX system calls. This information triggers the initialization of the user and machine models. An initialization file is read that resets the state of MDV to that of the last session. The scientist may then create a visualization specification for MDV. The visualization specification designates the subset of data to be viewed, the task, and the output medium. The scientist creates this specification using the MDV interface, shown in figure 4.1.

- New Dataset. If a new simulation has been generated, the scientist needs to register it with MDV. Registration is done using the New Dataset button. This action brings up an additional menu that will prompt the user for necessary information (attributes) about the simulation datasets. The scientist must select a new name for the dataset which will then be registered and stored under the Dataset Name menu.

- Dataset Name: The scientist must first select the dataset they are interested in viewing from the available datasets presented in the Dataset Name menu. A dataset represents the collection of data files associated with a simulation run. This could involve many files, depending on how the data is stored and what types of grids and physical variables are involved. This abstraction simplifies the otherwise tedious process of data input. A central repository stores the datasets. The data repository is simply a hierarchical directory system in which a single directory represents a dataset. Within this directory are all of the datafiles as well as a description file that contains the necessary metadata for that dataset. This description file is generated by asking the scientist several questions about the dataset when he/she enters it into the system. Future work will involve distributing this central repository so that larger datasets can be visualized.

- Fluids Object/Structures Object. Upon selection of a dataset, the scientist must select which components they are interested in viewing.

Figure 4.1: The MDV user interface.

This aspect of the visualization specification determines the subset of data to be visualized.

- Fluids Variables/Structures Variables. The scientist must select the physical variables he/she is interested in viewing. For example, if the scientist is interested in viewing the fluid flow around the deforming wing, he/she would select "Fluid Variable: Velocity" and "Structures Variable: Position".

- Task. The second component of the visualization specification is the selection of a data analysis task. Currently, the scientist may select from four tasks: *Debugging, Exploration, Analysis* and *Presentation*. These tasks were determined by working with scientists and understanding the main types of functions they were interested in performing. Based on these tasks, different visualization mappings and visual attributes are selected.

- Output. The machine model requires input regarding the preferred output medium so that the machine model can suggest an appropriate visualization mapping and visual attributes. The current options include: the screen, a color printer, or a black and white printer. The machine model contains the printer names of the nearest laser printer or color printer based on the workstation currently in use by the scientist.

- Timesteps. The scientist is presented with the range of timesteps that his/her dataset encompasses. The scientist must select all or a subset of these timesteps as part of the visualization specification.

- Visualize. Upon specification of these parameters, the scientist presses the Visualize button and an image is generated. The visualization generation cycle is performed and a visualization is presented.

- Current Visuals. This menu maintains a list of the visualizations generated by MDV during a working session. If the scientist is interested in viewing multiple visualizations simultaneously, the scientist can select the visualizations he/she is interested in and the visualizations will be drawn by the system.

- Long Menu. A more traditional interface is provided to allow experienced users to create visualizations directly.

- Exit. Upon completion of the data analysis session, the scientist can select this button to exit the system. The state of the system will be saved for the next session.

4.2.2 Visualization Mappings

The visualization mappings currently available in MDV are simple, yet effective, techniques for viewing fluid/structure interaction. Currently, the scientist can view wireframe and surface representations of the geometry of the simulation; scalar mappings such as contours and color contours; and also vector mappings such as vector plots, streamlines, and stream surfaces. Scientists can integrate quantitative information using line plots. These mappings are simple and intended for basic visualization needs. As scientists (and MDV) become more sophisticated, more advanced visualization mappings such as isosurfaces, cutting planes, vector field topology, volume rendering, and streaklines could be included.

4.2.3 Interactivity / Custom Widgets

Interactivity is an important aspect of exploring large datasets. In MDV, direct manipulation allows the scientist to change viewpoints by rotating, scaling and translating the visualization. The scientist can also override the selections made by the system. Visual attributes such as the color table, background color, *etc.* can be changed after the initial presentation is made. These changes are made using the options in popup menus, which appear when the scientist clicks the mouse button on a visualization (shown in figure 4.2).

Each popup menu is designed specifically for the visualization mapping appearing in the scene. Popup menus only contain menu options pertinent to the visualization that is presented. For example, if a contour is generated, the popup menu will contain functions that can modify the contour visual such as adjusting line width or changing the number of contour lines. The popup menu also presents the scientist with the data value that the selected point represents. The object-oriented paradigm is helpful in managing the popup menu specifications. Information from the data model, visualization mapping and viewing parameters is used intelligently to assemble the popup menus so they provide the necessary information.

4.2.4 Data Queries

Information contained in the data model can be extracted by the scientist by selecting an option on the popup menus associated with each visual. Once the scientist clicks on the visual of interest, a popup menu appears. They must select the "Data Info" option and attributes of the data are presented to them in a text window. This attribute information is stored in the data model.

Data may be queried at different levels within the data model hierarchy. The first example shows the result of a query to obtain information about

Figure 4.2: Popup menus in MDV.

the current working environment. This query is performed by clicking in the background of the viewing window and requesting MDV INFORMATION. The result is a description of the current information relevant to the rendered visualization. This information includes a basic description of the data, the user, the machine environment, and the visualizations currently rendered in the window.

```
mdv:
 data:
  datadir "/r/win58/u/wk/kmiceli/Data"
  dataset hsct/a-29
  time (6 120 6)
 user:
  username pramono
  domain struct
  level-of-expertise novice
  colormap "blue-red.map"
 machine:
  machine win58
  bw-printer im3
  color-printer color2
 visuals:
  fluid-visuals "Fluids PRESSURE COLORMAP ((10 142) (2 2) (1 15))"
  struct-visuals "Structures PRESSURE SURFACE"
```

The second example shows the result of a query to a specific component in the visualization. This query was performed by clicking a point of interest on the rendered visualization and requesting DATA INFORMATION. This query provides a more detailed description of the data represented by the visualization, including the data filenames, the timestep value, etc.

```
struct-data:
 components wing
 amplification 0
 data-fname "/r/win58/u/wk/kmiceli/Data/HSCT/A-29/qs_1.12"
 grid-fname "/r/win58/u/wk/kmiceli/Data/HSCT/A-29/grids_1.12"
 file-type q
 grid-type structured
 timestep 12
```

4.2.5 Explanation Facility

Along with each generated visualization is an explanation describing why the visualization was selected. This explanation facility allows the scientist to understand the reasons why the visualization and its corresponding attributes were selected. Scientists are more willing to accept (or reject) a presented visualization if they are aware of why it was selected.

This example shows the visualization mapping and why it was selected based on a given visualization specification. The task component of the specification was ANALYZE and the output medium was BLACK AND WHITE PRINTER. The query was performed by clicking on the generated visualization and selecting the VISUAL INFORMATION option on the popup menu.

```
Mapping:  CONTOUR-LINES
Reason:   Contour lines are an effective mapping for a quantitative
          analysis of the data.  Contour lines show trends and offer
          greater numeric precision.
          Black and White Contours are easiest to distinguish when
          output is a black and white laser printer.  Data trends
          in a colormapped visualization can be unclear when sent
          to a printer.
```

4.2.6 Long Menu

If the scientist rejects a visualization and prefers viewing the data in another manner, he/she has that option by selecting a button to return to a more complex, yet more functional interface. This interface was generated before the assistant-based system was created to test the functionality of the underlying system. This interface with all of its panels is displayed in figure 4.3.

4.3 Summary

This chapter presented the prototype MDV system and described its functionality and user interface. In addition, implementation issues were considered and trade-offs discussed.

The MDV visualization framework was implemented using the SuperGlue object-oriented visualization programming environment. The framework components were tightly integrated with visualization and data management functionality, which provided a conceptually simple environment for developing the prototype. Using the object-oriented model was very helpful in managing complex, heterogeneous datasets. Basic visualization primitives were available in a flexible and extensible environment. Adding information to the existing framework was simple due to both the object-oriented and interpreted nature of the environment. Additional factors in choosing the SuperGlue environment included the availability of on-site support and well as support for user-interface building tools.

Figure 4.3: The “Long Menu” in MDV.

However, there were several drawbacks to using the SuperGlue environment. First of all, SuperGlue did not have a structured knowledge-base or reasoning engine to handle the assumptions and design process. This drawback had potential affects on the design process, both in terms of performance and in effectively manipulating the knowledge base of information.

For the purposes of this work, SuperGlue was an effective tool. Since the contributions of the MDV system were to identify the information necessary for a visualization assistant, and to develop a user-tested prototype, a simplistic method of implementation was chosen. The complexity involved in integrating a knowledge-based system, visualization tools and potentially data management features was undesirable. For future projects interested in a larger, more robust system, the knowledge-based approach should be considered.

CHAPTER 5
DISCUSSION AND CONCLUSIONS

This chapter discusses user feedback gained throughout the design and development of the MDV system. Contributions of this thesis work are reiterated, followed by areas for future research and conclusions.

5.1 User Feedback

The main users of the MDV system were two scientists, a fluid dynamicist and a structural dynamicist, involved in performing the HSCT simulation on the Intel iPSC/860. Both scientists had previously used visualization to help them analyze their data. Before MDV was available, they used several different tools over the course of their careers to visualize their results. The tools allowed them to create plots and visualizations of data from their individual disciplines. However, they did not have any tool to visualize their disciplines simultaneously. As a result, they were very interested in the design and development of MDV.

Feedback from the scientists was gained through daily contact, demos and interviews. This information is presented in the next sections.

5.1.1 Daily Contact and Demos

The simulation scientists were located in the same building, which was helpful in establishing a strong communication link. Daily contact usually involved discussing topics such as the current state of the MDV project or the current state of the HSCT simulation effort. When discussing MDV, issues about data management, functionality, or interface design were typically discussed. When discussing the simulation effort, problems with the simulation software, interesting features of the simulated data, and implementation details were discussed. Discussions regarding the current state of the simulation software led to adjustments or additions to the MDV system to accommodate the scientists. For example, the scientists were interested in querying data values from the visualization, the functionality to click on a point of interest and retrieve data values was added. Informal discussion was often the catalyst for new functionality within the MDV system.

Demos were also given to the simulation scientists as well as other scientists and visualization researchers. This provided valuable information about the "look and feel" of the system. Input from those outside the project proved valuable, suggesting new ideas or questioning current methods.

5.1.2 Interviews

Interviews were performed at the initial stages of the design process and upon completion of the prototype. Initial interviews provided critical information about the data and the scientific domains that was necessary to build the visualization system. Sample questions and responses from these interviews are available in Appendix C. Final interviews were directed toward the prototype system and its user interface. Questions from the final interview are available in Appendix D with the scientist's response summarized in the following paragraphs.

5.1.2.1 Visualization

Scientists were first asked questions about visualization, its role in the data analysis process and the qualities that make a good visualization tool. Scientists stated that visualization was an important part of the simulation process. The scientists felt that visualization helped them to 1) debug their simulation codes, 2) understand the physics occurring in the simulation and 3) communicate the information in the simulation between themselves and their colleagues.

Scientists are interested in using visualization technology because they feel it is a helpful tool for data analysis. However, if a visualization system is complicated to use and comes with large manuals, scientists are less inclined to use it. Even online help is unattractive, except for precise answers to questions about system functionality. The initial steepness of the learning curve in using visualization systems often prevents scientists from using tools that could be very valuable to their research.

The important factors in using a visualization system are ease of use, functionality and performance. To use a system, the scientist must be able to sit down immediately and do productive work. Ease of use is the first attraction of a good tool and is especially important when the tool is new to a working environment. If a tool is difficult to use and productive work is not accomplished in a short amount of time, scientists will return to old methods and relay their experiences to colleagues.

As the scientist evolves from a novice to an expert user, the system functionality becomes more important, because the scientist wishes to perform more complex operations. Performance also becomes more important throughout this cycle, as the scientist will be uninterested in using a tool if the resulting visualizations take too long to generate. However, scientists are willing to wait short periods of time, if the resulting visualizations are worth the wait and provide a different insight into their data.

5.1.2.2 The MDV Visualization Assistant

When discussing the MDV system, the scientists were presented with the goal and basic concepts behind its development. The goal of the system was to make the visualization process more productive through the use of a knowledge-based visualization assistant. They were very responsive to the MDV design and agreed that if creating visualizations were an easier process, it would help them to be more productive. They were not visualization experts and they didn't intend to become experts to visualize their datasets.

Scientists currently must select from a "grab-bag" of visualization mappings. They often "make do" with tools they are familiar with, regardless of whether these tools express the information content they are looking for in their data. They believed that a system that helped them to select an effective and expressive visualization would remove the trial-and-error process they currently use.

Scientists were then asked specific questions about the MDV system. The scientists understood the complexity involved in multidisciplinary simulations, given the different data from different disciplines. This data management issue involved integrating many different files involved in a multiple timestep simulation as well as different data formats. The scientists stated that if they had to deal with data management themselves that they probably wouldn't use the visualization system. If data management could be handled by the visualization system, it would be much more attractive to use.

Perhaps the aspect of MDV that they appreciated most was the data management function incorporated into the DATASET NAME button. They were pleased with the data management aspect of MDV, which only required them to enter information about the dataset once. When referring back to that dataset, they selected the dataset alias representing the collection of files. This capability alleviated the tedious process of entering and managing the data files. The number of data files can be approximately one-hundred for a small simulation with forty timesteps. This aspect of the system alone would help increase productivity.

The scientists were generally comfortable about the querying mechanism that required them to input their data analysis task via high level abstractions. They were a bit confused by the process, since most of the tools they had used were much different from the MDV interface. Specifying the task they were interested in performing was a bit unusual to the scientist. They were not sure of the purpose of the TASK button until it was explained to them and an example was given. They said that the example made sense, but they were unsure of specific tasks they they attempted to accomplish while visualizing their data. For the most part they were just interested in viewing the data and querying it interactively.

The OUTPUT button was well received. The scientists agreed that knowledge about the output medium would be helpful in producing an expressive visualization. They also wanted the system to automatically generate relevant annotation and send the visualization to the appropriate device for printing. If the process of creating output could be automated, they felt it would save them time.

The automatic selection and generation of the visualization mapping was also a bit awkward to the scientist. They felt that having the system generate the first cut of the visualization was a good idea, especially when they were just exploring their data and wanted the visualization to be generated quickly. However, they were concerned that they might not be able to modify the visualization in case they were interested in viewing a different representation. They wanted access to the more functional interface readily available should this be necessary.

Custom widgets were provided to the scientist to query the data and give quantitative information to the otherwise qualitative images. The scientists were very pleased by this capability, allowing them to interact with their data and determine if their simulation was correct and if the data values made sense.

In general, MDV was well received by the scientists. They were able to test the interface and generate visualizations in a short amount of time. The user interface was not overwhelming and they could figure out how to use it almost immediately. They were happiest about the data management aspects of the system, relieved that they didn't explicitly have to input the large number of datafiles comprising the simulation. They also acknowledged that the help from the interface gave them the first step into viewing data from different disciplines, without having to know anything about visualization mappings within that discipline.

The scientists stated that a visualization assistant would make their data analysis process more productive, by helping them to generate effective and expressive visualizations in a short amount of time. They agreed that using MDV would help them produce results more quickly and easily than any other visualization tool.

The MDV system provided scientists with a tool to simultaneously visualize data from the fluids and structures domain. As far as we know, there is no other tool that can currently perform this task. Scientists relied on the images presented by MDV to help them study the results of their simulations. They often used the results they produced for presentations and publications [50].

5.1.2.3 Design Concerns

Some concerns were also uncovered by the scientists with regard to the MDV system and visualization systems in general. Some of these concerns were minor and involved easy-to-fix solutions. Other concerns questioned the overall design of the MDV system. Most of the minor concerns were handled during the design of MDV. These included concerns such as the layout of the system, the functionality of the system and the like. In addition, some knowledge about the simulation wasn't available to them. For example, information about where the actual geometry of the simulated vehicle was located within the three dimensional volume.

Some deeper concerns expressed by the scientists included a loss of functionality due to the limited capabilities of the visualization assistant. The scientists felt that the system would fit their needs in the early stages of data analysis, when they were unfamiliar with the system and didn't have clear idea of what they wanted to accomplish. In this scenario, the system would save them time and help them create effective visualizations. However, as they became more advanced users who wanted to accomplish more sophisticated data analysis goals, the visualization assistant might not be able to satisfy these needs and "get in the way".

Clearly, the visualization assistant approach is intended for users who are unfamiliar with tools and want to take advantage of the knowledge the system contains about visualization. However, as the scientist becomes as familiar with the system and visualization as the assistant, then the need for the assistant decreases. However, the assistant can still play a valuable role, albeit a lesser role. Rather than automatically selecting and generating a visualization, the system might only provide the expert with suggestions via a help window or something similar. The differentiation between novice and expert user could be incorporated into the user model and MDV would adjust its actions accordingly depending on the scientist's level of expertise.

Another issue of concern addressed by one of the scientists was that a complete knowledge-base of information could not be easily maintained on the simulated data. This concern stems from the disjoint three step simulation process that involves grid generation, field solution and post-processing data analysis (which includes visualization). Because the visualization aspect of this three step process is separate, knowledge gained from the two previous steps cannot be easily incorporated for use by the visualization system. This information includes special properties of the grid and the field solution that might be helpful during visualization. This high-level information is difficult to transfer to the visualization system.

A more advantageous approach to develop more productive tools for simulation scientists would be to incorporate the three steps of the simulation process into a single integrated system. Within this system, an assistant could guide scientists with the entire process and at the same time maintain all the design decisions involved throughout. However, this is a major undertaking involving a great deal of complexity and knowledge. The MDV visualization assistant takes a small step towards the development of such a system. The ideas generated by the MDV system could be applied in a larger scale system that incorporated grid generation and field solution. The concepts of data model, user model, and machine model would still hold for this larger scale system. The data model would contain information about the simulated geometry and all the associated data, complete with information describing special regions of the geometry of interest to the scientist. The user model would contain information about the user, and would influence the level of automation or the type of visualizations presented to the scientist. The machine model would contain information about the hardware available for computing field solutions, the field solution software available, hardware and software for visualization, etc. The knowledge base would contain information about the grid generation, field solution and visualization disciplines, as well as knowledge gained through the course of the design and simulation process.

5.1.2.4 Implementation Concerns

Other concerns expressed by scientists dealt with implementation issues, which do not directly reflect on the design of MDV, but did impact the scientists. The current implementation of MDV was performed on a single workstation. This workstation only had the ability to process 20 timesteps of the HSCT simulation. While this provided the scientist with enough information to study their results sufficiently, a system that was able to handle larger datasets would be beneficial. Future work may involve adapting MDV so that it could access the large datasets via a distributed file system. In addition, performing some of the computationally intensive work on a supercomputer would save the time spent on computing contour lines, etc on the workstation.

The base of users for feedback on the MDV system was small, but since the fluid/structure application was so specialized, not very many scientists were available. In addition, due to the participatory nature of the design process, it would have been difficult to interact with more than the two scientists. Input was solicited from other researchers not performing multidisciplinary simulations, although they were not users of the MDV system. Future efforts will include contacting other researchers in the field so that they might be able to take advantage of the software and provide additional feedback.

5.2 Contributions

Many visualization systems are complicated to use, with many commands, buttons and widgets. A great deal of research has been put into the development of new visualization mappings and a lot of work put into creating a general-purpose visualization tool. However, most of the focus in development these systems has bypassed the user interface. This research demonstrates a new style of interface to help scientists deal with the complexity of data analysis.

The contribution of this work comes in two parts, a high-level foundation and a user-tested implementation. In the design of the visualization assistant, the development of a structured framework is required to define the necessary components that must comprise such a system. The development of this framework provides a structure from which future systems can be built and expanded. This structured approach to visualization, as opposed to current *ad hoc* approaches, is a step towards understanding the visualization process as a whole.

The development of a visualization assistant helps create effective visualizations for the scientist. Effective visualization leads to more accurate interpretation of the data since knowledge from various domains such as graphic design and visual perception goes into the development of the representation. This knowledge is typically not possessed by the scientist who often relies on the help of a visualization expert. By freeing the scientist from having to consult with this expert, the visualization process can be made more productive. The visualization assistant approach is a significant contribution to the design of visualization systems in that it makes visualization technology more accessible to the typical scientist.

This research acknowledges the need to incorporate information from many different sources in order to create a useful visualization assistant. The information models and the attributes associated with them provided information regarding what factors influence the interpretation of visualizations. These attributes and their influence should be considered for any visualization task.

Although the visualization assistant was developed with multidisciplinary computational aerosciences data as the application, the framework and concepts in the framework could be applied to many different disciplines. Multidisciplinary visualization is necessary in many application domains, including medicine and environmental studies. The visualization programmer can learn from these ideas in developing new applications that provide the scientist with a productive and effective tool.

Data management in the visualization field has recently become recognized as a critical technology in the development of visualization tools [5]. As a result of the larger and more complex datasets that scientists are producing, visualization researchers have acknowledged the need for more sophisticated methods for data management. The object-oriented approach is a possible solution to the management problem. The object-oriented model is useful in modeling the data complexities, complete with syntactic and semantic information about the data.

The specific application chosen for this thesis was an excellent example of the complexity involved in analyzing current simulation datasets. The multidisciplinary nature of the application emphasized the need to develop tools that incorporated domain knowledge so that scientists of different disciplines could work with each other's data interchangeably. However, the visualization framework and the visualization assistant specified in this thesis could be applied to any application domain. The components of the framework are basic and central to anyone wishing to use visualization technology.

Although work has been done in encoding information about graphical representations, this project attempts to codify this information in such a way that it can be applied to visualization systems. The research presented in this thesis builds on the foundation set by the previous research projects mentioned in Chapter 1. It expands on this research by modeling the components that comprise the visualization process. As a result, the necessary information for generating effective graphics is available. This research emphasizes the importance of a data model and corresponding data attributes that influence the creation of effective and expressive visualizations.

By incorporating a data model, this work emphasizes data management and representation issues that have been missing from most visualization systems. The data model incorporates knowledge about the data as opposed to defining data structures that merely store the data. These abstractions allow the scientist to view the data in their own terms and not in the form of data structures defined by computer scientists. It also allows the system to develop effective visualizations based on this knowledge. Current systems lack data models that contain the high level knowledge that can assist the scientist with their data analysis. The object-oriented paradigm is used as the basis for the data model. The object-oriented paradigm allows for modularity, code reuse, expandability and the capability to derive data using methods.

This solution addresses the challenge of organizing and managing multiple, heterogeneous data sets and their resulting visual representations. Since multidisciplinary simulations will be more common as computing technology advances, this type of system must explore data management issues. Data management has finally been recognized by the visualization community as an essential component of any visualization system. This project attempts to address the data management issue by incorporating object-oriented data modeling techniques.

5.3 Future Research

MDV is under development and the current system is a very basic implementation of what is necessary to create a useful visualization tool for general use. A great deal of work has to be done in defining each component of the design framework, especially the data model. To develop this system to meet the given criteria, emphasis on the design of the framework is essential.

Future research plans include incorporating full user and machine models that take into account the task at hand as well as the varying types of input and output devices available to scientists. This effort will complete the characterization of the visualization environment and demonstrate the impact that each of these components has in the development of visualization systems. Finally, due to the large size of the data sets that will be encountered, database-oriented issues will need to be addressed. A distributed, persistent object database derived from the original data model would be valuable for managing large multidisciplinary data sets.

The further development of the user model and the machine model will help to include information about the scientists using the system as well as the specific machine requirements they have. The development of user models for user interface design is a research topic in its own right. This aspect of the system will undoubtedly require an iterative approach as well as close collaboration with application domain scientists.

Establishing the task the scientist is interested in performing is a critical component of the intelligent system. This aspect is currently being studied by interacting with the scientists and asking them to express their visualization goals. The generation of a task dialogue will eventually be integrated into the system, so that the scientist can specify which data they are interested in viewing as well as the goal of their data analysis task.

Finally, the continued addition of knowledge from the scientific domains, perception and graphic design will be an important part of future development efforts. These efforts will involve consulting with literature and with experts in the field in developing rules for effective data visualization.

Additional applications of this approach could be seen in the larger scale of the simulation process, assisting the scientist not only with the visualization aspect of the simulation, but also with generating the initial geometry and grids, selecting the appropriate solution scheme to solve the desired fields, and finally, aiding in the visualization of the results.

5.4 Conclusions

Visualization is becoming an increasingly complex tool for data analysis. This increase is due to the complexity of datasets, their increasing size and new research into tools and mappings. It is evident that this increasing complexity could drive away the most important customer of the visualization technology, the scientist. Often, developers of visualization tools do not consider usability issues when developing tools for use by scientists.

The development of easy-to-use visualization tools to deal with the complexity must be a critical concern to developers creating visualization tools. The key to this development is the abstraction of complexity using advanced modeling concepts and computer science technologies as seen in object-oriented design, databases, artificial intelligence and user interface studies. By taking advantage of these multiple disciplines in visualization, which is inherently a interdisciplinary research area, the success of these tools can be more easily accomplished.

The work in this thesis addresses this issue of abstracting complexity from the visualization process. The application chosen for demonstration is one that is probably on the high-end of the list in terms of complexity, so payoffs can be readily seen. However, the framework presented can be applied to any application in achieving a productive data analysis tool for scientists.

BIBLIOGRAPHY

[1] Robert M. Abarbanel and Michael D. Williams. A relational representation for knowledge bases. In Larry Kerschberg, editor, *Expert Database Systems*. The Benjamin/Cummings Publishing Company, 1987.

[2] Zahid Ahmed, Kristina Miceli, Steve Casner, and Steve Roth, editors. *Workshop on Intelligent Visualization Systems*. IEEE Computer Society, Visualization '93, October 1993.

[3] Zahid Ahmed, Len Wanger, and Peter Kochevar. An Intelligent Visualization System for Earth Science Data Analysis. In Zahid Ahmed, Kristina Miceli, Steve Casner, and Steve Roth, editors, *Workshop on Intelligent Visualization Systems*. IEEE Computer Society/Visualization '93, October 1993.

[4] Gordon V. Bancroft, Fergus J. Merritt, Todd C. Plessel, Paul G. Kelaita, R. Kevin McCabe, and Al Globus. FAST: A Multi-Processed Environment for Visualization of Computational Fluid Dynamics. In Arie Kaufman, editor, *Proceedings of Visualization '90*, pages 14–24. IEEE Computer Society, 1990.

[5] R. Daniel Bergeron, William Cody, William Hibbard, David T. Kao, Kristina D. Miceli, Lloyd A. Treinish, and Sandra Walther. Database Issues for Data Visualization: Developing a Data Model. In John P. Lee and Georges G. Grinstein, editors, *Database Issues for Data Visualization*. Springer Verlag, 1994.

[6] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, Madison, WI, 1983.

[7] Clifford Beshers and Steven Feiner. Autovisual: Rule-based design of interactive multivariate visualizations. *IEEE Computer Graphics and Applications*, 13(4), July 1993.

[8] David M. Betz. Xscheme: An object-oriented scheme. (URL http://cs.indiana.edu/pub/scheme).

[9] Jeanette L. Blomberg and Austin Henderson. Reflections on participatory design: Lessons from the trillium experience. In *Proceedings of CHI '90*, pages 353–359. ACM/SIGCHI, May 1990.

[10] Grady Booch. *Object-Oriented Design with Applications*. The Benjamin/Cummings Publishing Company, Redwood City, CA, 1991.

[11] Ronald J. Brachman and James G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. In Ronald J. Brachman and Hector J. Levesque, editors, *Readings in Knowledge Representation.* Morgan Kaufmann Publishers, 1985.

[12] D.M. Butler and M.H. Pendley. A Visualization Model Based on the Mathematics of Fiber Bundles. *Computers in Physics*, 3:45–51, 1989.

[13] B. Cabral and C. Leedom. Imaging vector fields using line integral convolution. *Computer Graphics*, pages 263–270, August 1993.

[14] Steven M. Casner. A Task Analytic Approach to the Automated Design of Graphic Presentations. *ACM Transactions on Graphics*, 10(2):111–151, April 1991.

[15] R.G.G. Cattell. *The Object Database Standard: ODMG-93, Version 1.1.* Morgan Kaufmann, 1993.

[16] W.S. Cleveland and R. McGil. Graphical Perception: Theory, Experimentation, and Applications to the Development of Graphical Methods. *Science*, pages 828–833, August 1985.

[17] Thierry Delmarcelle and Lambertus Hesselink. Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics and Applications*, 13(4), July 1993.

[18] Michael Dertouzos. Modern Interfaces for Ancient Humans, 1992. Presentation at the MIT Media Lab Symposium on Interface Agents, October 20, 1992.

[19] Gitta Domik. The Role of Visualization in Understanding Data. In *Lecture Notes on Computer Science 555: New Trends and Results in Computer Science*, pages 91–107. Springer Verlag, 1991.

[20] Gitta Domik. A Paradigm for Visual Representations. Technical report, University of Colorado, Computer Science Department, CB 430, Boulder, CO 80309-0430, 1992.

[21] G.O. Domik and B. Gutkauf. User modeling for adaptive visualization systems. In *Proceedings of IEEE Visualization '94*, 1994.

[22] S. Feiner and C. Beshers. Visualizing n-dimensional virtual worlds with n-vision. *Computer Graphics*, 24(2), March 1990.

[23] Steven Feiner, Jock Mackinlay, and Joe Marks. Automating the Design of Effective Graphics for Intelligent User Interfaces. In *Tutorial at the 1992 Conference on Computer Human Interaction*, 1992.

[24] G. Fischer. The Importance of Models in Making Complex Systems Comprehensible. In D. Ackerman and M. Tauber, editors, *Mental Models and Human Computer Communications*, pages 3–36. Elsevier Science, 1991.

[25] Gerhard Fischer, Andreas C. Lemke, Thomas Mastaglio, and Anders I. Morch. The Role of Critiquing in Cooperative Problem Solving. *ACM Transactions on Information Systems*, 9(3):123–151, April 1991.

[26] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice, Second Edition.* Addison-Wesley Publishing Company, 1990.

[27] Al Globus, Creon Levit, and Tom Lasinski. A Tool for Visualizing the Topology of Three-Dimensional Vector Fields. In Gregory M. Nielson and Larry Rosenblum, editors, *Proceedings of Visualization '91*, pages 33–40. IEEE Computer Society, 1991.

[28] Jade Goldstein, Steven F. Roth, John Kolojejchick, and Joe Mattis. A Framework for the Automatic Design of Large Data Set Displays. In Zahid Ahmed, Kristina Miceli, Steve Casner, and Steve Roth, editors, *Workshop on Intelligent Visualization Systems.* IEEE Computer Society/Visualization '93, October 1993.

[29] B. Gutkauf. User modeling in scientific visualization. Master's thesis, University of Colorado, Boulder, 1994.

[30] Robert B. Haber and David A. McNabb. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In G.M. Nielson and B. Shriver, editors, *Visualization in Scientific Computing.* IEEE Computer Society Press, 1990.

[31] R.W. Hamming. *Numerical Methods for Scientists and Engineers.* McGraw-Hill, New York, 1962.

[32] T.L. Holst, M.D. Salas, and R.W. Claus. The NASA Computational Aerosciences Program Toward Teraflops Computing. In *30th Aerospace Sciences Meeting and Exhibit*, 1992.

[33] J.P.M. Hultquist. Numerical Flow Visualization in a Functional Style. Technical Report RNR-89-008, NASA Ames Research Center, MS T27A-1, Moffett Field, CA 94035, June 1989.

[34] J.P.M. Hultquist. Constructing Stream Surfaces in Steady 3D Vector Fields. In Arie E. Kaufman and Gregory M. Nielson, editors, *Proceedings of Visualization '92*, pages 171–178, 1992.

[35] J.P.M. Hultquist and E.L. Raible. Superglue: A Programming Environment for Scientific Visualization. In Arie E. Kaufman and

Gregory M. Nielson, editors, *Proceedings of Visualization '92*, pages 243–251, 1992.

[36] R.W. Hunt. *Measuring Color.* Market Cross House, England, 1987.

[37] D. Kahneman and A. Henik. Perceptual Organization and Attention. In M. Kubovy and J.R. Pomerantz, editors, *Perceptual Organization*, pages 181–211. Lawrence Erlbaum, 1981.

[38] Robert Kass and Tim Finin. General User Modeling: A Facility to Support Intelligent Interaction. In Joseph W. Sullivan and Sherman W. Tyler, editors, *Intelligent User Interfaces.* ACM Press, 1991.

[39] Peter R. Keller and Mary M. Keller. *Visual Cues.* IEEE Computer Society Press, 1993.

[40] Won Kim. *Introduction to Object-Oriented Databases.* The MIT Press, 1990.

[41] David A. Lane. Visualization of Time-Dependent Flow Fields. In Gregory Nielson and Dan Bergeron, editors, *Proceedings of Visualization '93*, pages 32–38. IEEE Computer Society, 1993.

[42] Bruce Lucas, Gregory D. Abram, Nancy S. Collins, David A. Epstein, Donna L. Gresh, and Kevin P. McAuliffe. An Architecture for a Scientific Visualization System. In Arie E. Kaufman and Gregory M. Nielson, editors, *Proceedings of Visualization '92*, pages 107–114, 1992.

[43] Jock Mackinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2):110–141, April 1986.

[44] Aaron Marcus. *Graphic Design for Electronic Documents and User Interfaces.* ACM Press, 1992.

[45] Bruce H. McCormick, Thomas A. DeFanti, and Maxine Brown. Visualization in Scientific Computing. *Computer Graphics*, 21(6), November 1987.

[46] Gary W. Meyer and Donald P. Greenberg. Color-Defective Vision and Computer Graphics Displays. *Computer Graphics and Applications*, 8(5):28–40, September 1988.

[47] Kristina D. Mickus. Participatory User Interface Design for Scientific Visualization Systems. Master's thesis, University of Colorado, Boulder, 1991.

[48] National Center for Supercomputer Applications, Champaign, IL. *NCSA HDF Calling Interfaces and Utilities, Version 3.0*, 1989.

[49] John Ousterhout. *The Tcl/Tk Toolkit.* University of California at Berkeley, 1993.

[50] Eddy Pramono and Sisira Weeratunga. Aeroelastic Computations for Wings Through Direct Coupling on Distributed-Memory MIMD Parallel Computers. In *32nd AIAA Aerospace Sciences Meeting*, Reno, NV, January 1994. AIAA Paper No. 94-0095.

[51] Lori Resnick. *CLASSIC Description and Reference Manual for the Common LISP Implementation.* AT&T Bell Laboratories, Murray Hill, NJ, 1993.

[52] R.K. Rew and G.P. Davis. NetCDF: An Interface for Scientific Data Access. *Computer Graphics and Applications*, July 1990.

[53] Philip K. Robertson. A Methodology for Scientific Data Visualization: Choosing Representations Based on a Natural Scene Paradigm. In Arie Kaufman, editor, *Proceedings of Visualization '90*, pages 114–123. IEEE Computer Society, 1990.

[54] Philip K. Robertson, Lisa De Ferrari, Peter Fletcher, and Guy Vezina. Visualisation on a Massively Parallel Supercomputer. Technical Report TR-HJ-91-04, CSIRO Division of Information Technology, Center for Spatial Information Systems, GPO Box 664, Canberra, ACT 2601, Australia, 1991.

[55] Bernice E. Rogowitz and Lloyd A. Treinish. An Architecture for Rule-Based Visualization. In Zahid Ahmed, Kristina Miceli, Steve Casner, and Steve Roth, editors, *Workshop on Intelligent Visualization Systems.* IEEE Computer Society/Visualization '93, October 1993.

[56] Lawrence J. Rosenblum. Research Issues in Scientific Visualization. *IEEE Computer Graphics and Applications*, pages 61–85, March 1994.

[57] Steven F. Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. Interactive Graphic Design Using Automatic Presentation Knowledge. In *Proceedings of CHI '94.* ACM SIGCHI, April 1994.

[58] Steven F. Roth and Joe Mattis. Data Characterization for Intelligent Graphics Presentation. In Jane Carrasco Chew and John Whiteside, editors, *Empowering People, CHI 1990 Conference Proceedings*, pages 193–200. ACM SIGCHI, 1990.

[59] Paolo Sabella and Ingrid Carlbom. An Object-Oriented Approach to the Solid Modeling of Empirical Data. *IEEE Computer Graphics and Applications*, September 1989.

[60] Hikmet Senay and Eve Ignatius. Rules and Principles for Scientific Data Visualization. Technical Report GWU-IIST-90-13, George Washington

University, Institute for Information Science and Technology, Department of Electrical Engineering and Computer Science, School of Engineering and Applied Science, Washington, D.C., 20052, May 1990.

[61] Hikmet Senay and Eve Ignatius. VISTA: Visualization Tool Assistant for Viewing Scientific Data. In *SIGGRAPH 1990 Course Notes: State of the Art in Data Visualization*, 1990.

[62] Hikmet Senay and Eve Ignatius. A Task-Oriented Approach to Scientific Visualization Using Perceptual Dimensions and Organization. In *IEEE Visualization '92 Workshop on Automated Design of Visualizations*, October 1992.

[63] Silicon Graphics Corporation, Mountain View, CA. *Explorer User's Guide, Module Writer's Guide*, 1991.

[64] Joseph W. Sullivan and Sherman W. Tyler, editors. *Intelligent User Interfaces.* ACM Press, 1991.

[65] L.A. Treinish and M.L. Gough. A Software Package for Data Independent Management of Multidimensional Data. *EOS Transactions, American Geophysical Union*, 68, 1987.

[66] E. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Chesire, CT, 1983.

[67] E. Tufte. *Envisioning Information.* Graphics Press, Chesire, CT, 1990.

[68] Craig Upson. The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE Computer Graphics and Applications*, 9(4), July 1989.

[69] Samuel P. Uselton. Volume Rendering for Computational Fluid Dynamics: Initial Results. Technical Report RNR-91-026, NASA Ames Research Center, MS T27A-1, Moffett Field, CA 94035, September 1991.

[70] Stephen Wehrend and Clayton Lewis. A Problem-Oriented Classification of Visualization Techniques. In Arie Kaufman, editor, *Proceedings of Visualization '90*, pages 139–143. IEEE Computer Society, 1990.

APPENDIX A
SELECTED RULES FROM PERCEPTION AND GRAPHIC DESIGN

Basic rules have been incorporated into MDV on the use of color in displays and for visualization purposes. These rules were obtained from [19, 39, 44, 60], who in turn gathered their information from many different sources.

A.1 Color Perception

1. Colormaps may carry unintended meanings; therefore, use colormaps with care. It is best to use familiar colormaps. For example, astronomers use color scales where red represents faint stars (low values), yellow represents brighter stars (medium values), and white represents the brightest stars (high values). This is different from Earth scientists and geologists who use color scales where blue represents oceans (low values), green represents forests and meadows (medium values), and brown represents mountains (high values). When in doubt, use spectral ordering in colormaps; viewers see spectral ordering as a natural one. Greyscale is also another good choice as it has a natural visual order.

2. Use additional cues to delineate shape, such as contour lines, when colormapping scientific data. Outlining the colormapped data helps the eye to distinguish between two colors.

3. Use a sudden color change to mark a critical level. The abrupt change of color signals the preattentive vision of the viewer. Preattentive vision is the instantaneous, effortless part of visual perception without focusing on local detail[16].

4. For presentations in a dark room: select sharply contrasting colors. Background dark colors such as dark blue appear to recede, and yellow or white text stands out and provides contrast. Pastels and bright colors are good choices for data. For presentations in a light room, use lighter colors for the background and darker colors for text.

5. Be careful in displaying colors of low intensity (*i.e.* blue and black) close together if a distinction between the two colors needs to be made. It is difficult to distinguish between these colors. The same rule applies with high intensity (*i.e.* white and yellow) colors close together.

6. Because cones that detect short wavelengths lie in outer regions of the fovea, blue (short wavelength) is a good color to use in the periphery of a display. However, blue is a poor choice for the display of small objects because the lack of short wavelength cones in the fovea. Other good colors to use in the periphery include black, white, and yellow. Red and

green are good colors to use in center of visual field, due to the greater number of red and green cones in the center of the fovea.

7. Use perceptual color scales such as Hue, Lightness, Value (HLS), Hue, Saturation, Value (HSV), or CIELUV (perceptually uniform color space [26]) instead of RGB to aid with perception.

8. Color in the interface should be used carefully. Confusion might result if color is used to group menu items and also display elements.

A.2 Graphic Design

1. Avoid using color that does not add or support meaning of data.

2. Greyscale and black and white encourage simplicity and remove the complications that may arise due to the physiological, psychological, or printing side-effects of color.

3. Lines are helpful to show connection and relationships. They separate, highlight, focus, and indicate direction. Increased line thickness is typically an indication of importance.

4. A general principle for annotation is to use little, but make it meaningful. The text size in annotations should vary according to the importance of the object it identifies. The text style should be simple, legible and uniform.

5. If the intended audience is broad, use a less complex representation. Simple images are best for communication. If the audience is well-informed, more complex images may be used.

6. Use horizontal, or landscape, orientation if possible. It is often preferred because it corresponds to the normal field of vision.

APPENDIX B
SELECTED RULES FROM SCIENTIFIC DOMAINS

B.1 Fluid Dynamics

The multidisciplinary simulation will help fluid dynamicists to understand how the deforming body of the HSCT affects flow characteristics. Characteristic visualization mappings and attributes were gathered from literature and interaction with scientists.

1. Data is commonly colormapped using the spectral color scale with color ranging from red (high) to yellow (freestream conditions, average conditions) to blue (low).

2. Scalar velocity magnitude is colormapped on the surface using the color scheme presented above. Velocity is also often colormapped in terms of its x,y, and z components.

3. Several gridplanes are often depicted so that a relationships along the object body can be established.

4. In order to emphasize shock structure, black contours are drawn on a colormapped visualization of vorticity

B.2 Structural Dynamics

The structural dynamicist is interested in understanding the deformation of HSCT and the resulting stress levels on the skin of the vehicle. Presented below are some of the visualization mappings that are helpful to the structural dynamicist.

1. Data is commonly colormapped using the spectral color scale.

2. Colormapping and contour lines are most frequently used to show pressure and stress levels.

3. Animation is useful to show the deformation of the vehicle over time.

4. Plots are helpful to show deformation of a point on the grid over time.

APPENDIX C
INITIAL USER INTERVIEW: DATA CHARACTERISTICS

Initial user interviews were held to understand the data that was to be modeled in the multidisciplinary visualization system. In addition, questions were asked regarding the type of tools that would be necessary for multidisciplinary data analysis. The next section is a user questionnaire presented to the scientist. Following the questionnaire is a response from a structural dynamicist.

C.1 Questionairre

1. Narrative description of data type (coordinate system, special proporties of the mesh, any information which explains the data type and the reason why data are stored in specific structures)
2. Definition of terms
3. Size information (how large is a typical data set?, how many data sets?, number of iterations?)
    - How much of the data is fluids, how much structures?
    - In what format is data stored?
4. Variable descriptions
    - Variables which describe the mesh (structured or unstructured grids, moving grids, new grid per timestep?)
    - Physics variables
    - Attribute data (size, shape, rank, conditions of simulation)
5. Methods (how will you want to access your data, methods for operating on the data)
6. Graphics and usage requirements
7. Other questions
    - What information are you trying to extract/what are you looking for in your data?
    - What characteristics does your data have?
    - What types of visualization mappings do you use to view your data?
    - Do you view the entire dataset or just parts of it?
    - How does your data affect other disciplines in a multidisciplinary simulation?

- Will you have to look at data from other disciplines?
- Do you know anything about the other discipline and its data?
- How will it effect your data?
- How do you want to interact with the data?
- What relationships will you be interested in?
- What type of interface would you like?
- What kind of capabilities would you like the system to have?
- How does the simulation work?
- How are quantitative/statistical techniques beneficial to data analysis?
- Would you like to visualize the data as the simulation progresses? How do you envision this happening?

C.2 User Response

The following is the response from a interview held with a structural dynamicist regarding the development of visualization tools. The responses are brief comments that were taken during the course of the interview.

1. Narrative description of data type (coordinate system, special properties of the mesh, any information which explains the data type and the reason why data are stored in specific structures)

    Multidisciplinary goal: High Speed Civil Transport, unstructured grid for structures part of the code, fluids grid is structured. Structures is single grid.

    Data includes node information in world coords (X,Y,Z), topology, stress (scalar) and deformation (vector)

2. Definition of terms

    Deformation, displacement - vector of new X, Y, Z positions, not change in X, Y, Z direction

    Stresses - scalar value, one per timestep (you can specify stress in X, Y, Z direction prior to performing simulation, but does not make sense to visualize more than one stress variable at a time)

3. Size information (how large typical data set, how many data sets, iterations)

    Example - 9,000 pts, 1000s of timesteps

4. Variable descriptions

   - Variables which describe the mesh (structured/unstructured grids, moving grids, new grid per timestep?)

     - Variables which describe the mesh

       X, Y, Z - real world coordinates
       Topology - connection information for unstructured grid

     - Variables which describe material in each zone

       Can have different materials with different properties, but not common

     - Physics variables

       Deformation - X, Y, Z world coordinates
       Stress - scalar value, magnitude of stress at given node

     - Miscellaneous data, such as time step, cycle number

       Timesteps, can be 1000s of them

5. Graphics and usage requirements

   Capability to print picture on screen

6. Other questions

   - What information are you trying to extract/what are you looking for in your data?

     Stress levels in skin of vehicle. Looking for material failure also deformation, even though stress is typically more important.

   - What types of visualization mappings do you use to view your data?

     Stress - use color maps with capability of changing range of color tables based on data.
     Deformation - amplification factor and then animation to see how deformation progresses.
     Plots - deformation at a point vs time, want to see where flutter begins and how it progresses.
     Debugging tool - look at error values to determine if code is working correctly.

   - How does your data affect other disciplines in a multidisciplinary simulation?

     Deformations will affect fluid interaction by changing the shape of the vehicle.

   - Will you have to look at data from other disciplines?

Yes

- Do you know anything about the other discipline and its data?

  No

- How will it effect your data?

  To see how pressure and stress/deformation relate.

- What relationships will you be interested in?

  Deformation/pressure,
  Stress level history.

- What type of interface would you like?

  Point and click, ability to interactively mouse around and get data values.

- What kind of capabilities would you like the system to have?

  Ability to narrow down region of interest.
  Changing point of view, zoom, control color range, amplitude, cutting planes to see deformation.
  Ability to display while progressing through computation.

APPENDIX D
FINAL USER INTERVIEW: VISUALIZATION ASSISTANT

Final user interviews were held upon completion of this thesis project. The two scientists who most frequently used the system were questioned regarding their impressions of the MDV system. The next section contains the questionnaire. User feedback from these interviews was summarized in Section 5.1.

D.1 Questionairre

Previous Tools

- How important is visualization to you? How often do you use it as a tool?
- Current use of Visualization Software
  - What other tools do you currently use? Do you use it often?
  - What are the good points about it?
  - What are the bad points about it?
- What is most important to you in a tool for viewing your data?
  - Functionality (lots of different representations),
  - Ease of Use (easy to generate a visualization),
  - Performance (rapid generation),
  - Other

MDV

- Is the interface easy-to-use? Confusing? What are your first impressions?
- Does the high level of abstraction help when creating a visualization?
- Is it more natural to generate a visualization in this manner?
- Do you dislike letting the machine select the visualization technique and viewing parameters?
- Is functionality compromised by the high level of abstraction? Does this affect the way you work?
- What are your thoughts on the "Tasks" ? Do they seem appropriate?
- Is this a natural way to specify what you are try to get out of the data?

- Do they help or only hinder the process?
- What are your thoughts about the Output button? Is this helpful?