Proceedings of DETC 2000
2000 ASME Design Engineering Technical Conferences
September 10-13, 2000, Baltimore, Maryland

DETC2000/DAC-14261

USING A MULTIPLE CONCURRENT DESIGN VIEWS INTERFACE TO ENHANCE DESIGN COMPLEXITY MANAGEMENT

Benjamin A. Baker, Russell D. Fish and Elaine Cohen
Department of Computer Science
University of Utah
50 South Central Campus Drive, Room 3190
Salt Lake City, Utah 84112
Phone: (801) 581-8224
Email: bbaker@cs.utah.edu, fish@cs.utah.edu, cohen@cs.utah.edu

ABSTRACT

Modern product design complexity is a problem faced by designers of complex geometric products. It is very difficult for a designer to assimilate the vast amounts of data necessary to produce and understand such complex designs in their totality. Recently, research has been done to better manage design complexity, but little or no work has been done with respect to user interfaces for complexity management tools.

In this research, we present a concurrent design views interface to enhance design complexity management. This user interface assists the designer in creation and visualization of complex design frameworks. These design views support different levels of design detail and complexity, and also provide hierarchical design decomposition of complex design frameworks. Concurrency between design views is maintained, thus increasing the overall power of the system. To demonstrate the validity and applicability of this approach in solving complexity management issues, a prototype system implementation of an intuitive user interface built upon an existing complexity management framework is presented.

INTRODUCTION

Due to the complexity of modern product design, there exists a need to facilitate creation and visualization of complex design structures. In this research, we present a multiple concurrent design views interface to increase the designer's ability to manage these complex design structures, thus increasing the productivity and ease of use of 3D CAD systems. We use the term *view* or *design view* in this research to refer to a particular representation of the design framework. In particular the aim of these representations or design views is to assist the designer in assimilating the great amounts of data necessary to create, navigate and visualize complex design structures.

When designing a complex geometric model, developers often first create a high-level design of the product. This often consists of drawing a box diagram of the basic components of the system, broken up into a number of smaller sub-problems which can be more easily managed. Once a reasonable high-level design has been achieved, additional details may be added to this high-level conceptual design. Therefore, the high-level design represents the basic framework of a product, so it would be very advantageous for the designer to be able to use this framework within the CAD system itself.

In addition to creation of complex model structures, it is also difficult to navigate and visualize these model structures when using a conventional 3D CAD system. The hierarchical nature of a model framework is very suitable to being represented in a hierarchical browser. This added functionality would allow the designer to quickly navigate through a complex model structure.

Each of the methods described above provides additional capabilities to a product designer, however, for the ultimate power of these design views to be realized, it is necessary to maintain concurrency between all model design views. In other words, changes made in one view should be reflected in other views as well. In addition any feedback to the user should be consistent through all design views, thereby increasing user-interface consistency.

BACKGROUND AND RELATED WORK

Even though many design tools have been developed to automate difficult and time-consuming design tasks, little work has been done to include user interface capabilities with

complexity management tools. It is true the many 3D CAD systems contain user interface tools for geometry creation, but most do not provide user interface support throughout the product development cycle. The conceptual design stage is particularly devoid of such interfaces. The lack of a simple user interface that is able to integrate all parts of the product development cycle requires the designer to have specialized knowledge of certain complex design tools. This decreases the usability and ease of use of any complexity management tool, regardless of its innate power.

The work of Zanella and Gubian recognizes the importance of a user interface in complexity management, presenting the idea of a design manager (1996). They define a design manager as everything in the user interface of a CAD system that helps the user develop a product from conceptual design to final implementation. However, their research only made general considerations for developers of design manager systems, but presented no actual design manager system implementation.

Gorti and Sriram assert that existing methods for geometric product design have had little impact on the conceptual design stage of a product (1996). They propose a conceptual design agent (CONGEN) as a method of mapping from the symbolic nature of a conceptual design to the form of a geometric definition. While CONGEN provides a user interface for this symbol to form mapping which is useful for visualizing high-level concepts at the conceptual design stage, it can not easily be generalized to accommodate future more detailed geometry.

Feng et al. developed a representation of functions and features that deals with conceptual design issues, realizing that most of the total life cycle cost of a product is determined at the conceptual design stage (1996). However, the approach of their research concentrates on a graph representation of components of a complex system. No system implementation is presented and no discussion of how the designer would interact with this representation is presented.

Rosenman and Gero use different design views of the same primitive design objects to provide necessary domain information to specific persons involved with different portions of the design process (1996). While this approach is commendable, specific tools and interfaces suited for particular design tasks are missing from their implementation. Instead the multiple views are simply instances of a previously defined object.

Loss presents an interactive design views environment to provide support throughout the entire design process, particularly in the early stages of conceptual design (1997). His research makes connections between abstract design information and detailed 3D design views. Specifically, the idea of a *design web* is presented which allows distributed development of a product. Consideration is given to abstract information such as product specifications and design constraints. Our work is intended to build upon this idea of complete design process support by providing additional tools to connect processes at various stages of the overall design process.

The research of Jacobs provides a comprehensive organizational framework for representing, analyzing and controlling complex design models as they evolve (1998a). The basis for this framework uses three main types of components: *aggregation, relationship* and *version objects.*

An *aggregation object* encapsulates multiple design components to form a single entity (Jacobs, 1998b). Aggregations may be nested within other aggregations, thus forming a model hierarchy. *Relationship objects* provide methods for linking aggregations together. They include methods for analyzing and validating relationships between design components. In addition, *version objects* capture changes to aggregation and relationship objects, thus recording the design alternatives and history of a design as it evolves.

Jacobs' research provides comprehensive solutions to many complexity management problems. A more complete analysis of the complexity management capabilities of various systems is also presented in his work. Each of these design tools, including the work of Jacobs himself, provides solutions to portions of the complexity management problem. However, each implementation contains little or no user interface capabilities to aid the designer in using the design tool that was developed.

Even though the work of Jacobs in this area is very comprehensive, he realized the importance of a user interface to further extend the power of his design framework. In the suggestions for future work section, he suggests a hierarchical browser would significantly improve the designer's ability to visualize and navigate the complex structures of a modern design product (1998a).

Our research presents such a hierarchical browser, capable of hierarchical design decomposition. In addition, a 2D design view capable of supporting multiple levels of design detail and complexity has been added to further strengthen the complexity management capabilities of the overall system. By so doing, the problem of visualization and navigation of geometric product design hierarchies is solved through the use of a powerful user interface built upon existing complexity management components.

CONCURRENT DESIGN VIEWS

Various mechanisms and tools have been developed by researchers to manage complex design frameworks of modern CAD products. However, most do not include user interfaces that aid the designer in managing design complexity, so powerful mechanisms for managing design complexity are reduced to being used by a few developers with experience and skills with specialized mechanisms.

The concurrent design views interface incorporates an easy to use interface with design complexity management features to produce a superior integrated system that manages design complexity quite well. The concurrent design views

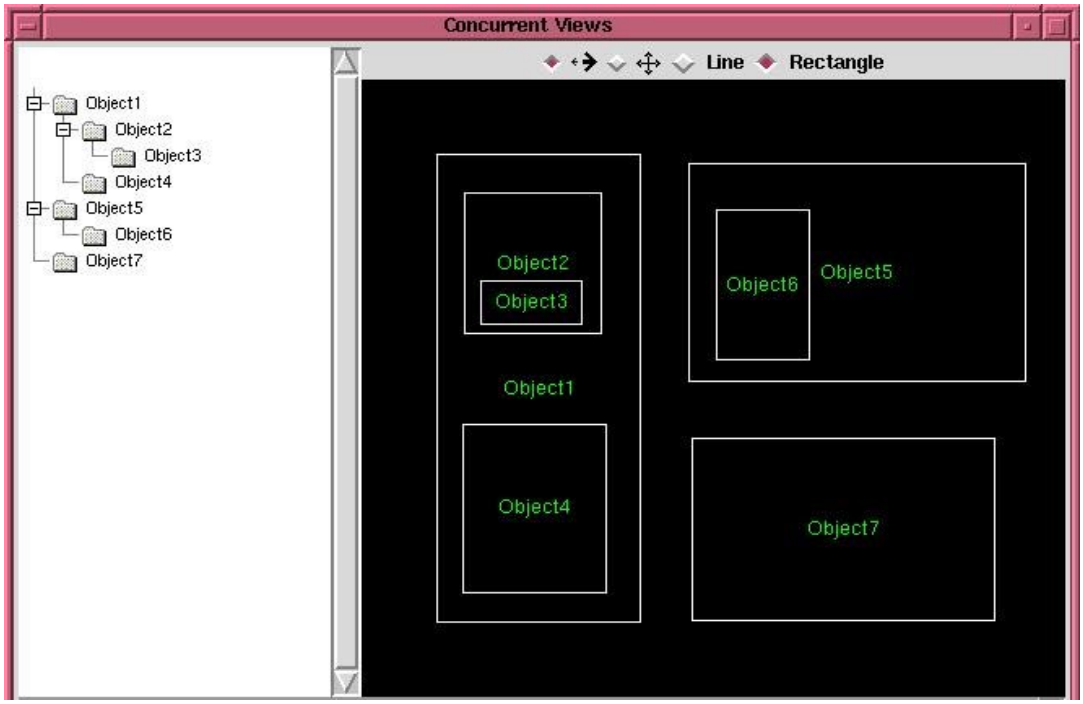

Figure 1. The concurrent design views interface

interface supports multiple levels of design detail and complexity as well as hierarchical decompositions of model structures. In addition, design alternatives and histories are supported by this system.

The concurrent design views interface augments an existing 3D CAD system which already supports many design features. It is designed to use the framework described by Jacobs in the previous section (1998a). We also adopt the definitions of aggregation, relationship and version objects used in his research.

The concurrent design views interface is comprised of two main components: the 2D Aggregation View and Hierarchical Tree View. (Figure 1) These views serve as tools aiding the designer with visualization and analysis of complex design frameworks. Concurrency between data displayed in each view is always maintained. This important feature allows the concurrent design views interface to be used throughout the production cycle of a product design.

2D Aggregation View

In order to develop a complex product in a modern CAD design system, it is necessary to first break up the design into smaller sub-problems. Designers usually develop some sort of high-level conceptual design, possibly a box diagram, indicating the main components of the system as well as interactions between these components. Then portions of the problem are assigned to separate individuals or groups to be solved.

The components developed by Timothy Jacobs are well suited to create a high-level conceptual design of a product and will be used as the underlying framework for design decomposition. The 2D Aggregation View uses rectangles as the representation of aggregation objects. When created, these aggregations serve as placeholders in which future detail design can be placed. Relationship objects are represented by lines connecting aggregation objects. These represent the interfaces between design components. Therefore, the 2D Aggregation View allows the designer to create and edit a high-level box diagram of the product while simultaneously generating the framework needed for future detail design.

As stated earlier, aggregations are objects that can contain multiple design components. They may also be nested, thus providing multiple levels of abstraction. The 2D Aggregation View allows for these multiple levels of abstraction through support of scaling and panning of the aggregation views. By doing this, very complex models may be decomposed into smaller parts and an infinite number of levels of design abstraction are possible.

Hierarchical Tree View

Complex design frameworks are by their very nature hierarchical in structure. This is due in part to the decomposition at multiple levels of detail of the description of a geometric object. Therefore, it is both intuitive and advantageous for a CAD environment to have a hierarchical structure browser to represent objects in the design framework.

So, by adding a hierarchical browser to the CAD system, the designer has the ability to quickly view and analyze the overall structure of a design. This also facilitates rapid navigation of the model hierarchy, thereby providing the

designer with the ability to quickly change levels of abstraction as well as perform operations on certain parts of the overall system.

The Hierarchical Tree View supports these operations by using a tree structure to represent the design framework. This interface works similar to file directory tree structure interfaces that are familiar to most designers. Therefore, the intuitive nature of this interface serves to enhance the power of this design view to represent complex design frameworks.

Concurrency Between Views

Often when a new tool is added to a CAD system, it supports only a specific new function. While it may support this task very well and the task may be very useful to the designer, if communication between components of the system is not maintained, the overall power of the system is lost. For example, if a tool is created to aid the designer with conceptual design, yet requires reentry of data already entered into the system when actually developing the product, then the conceptual design tool provides no more functionality than separately designing and implementing a product.

The concurrent views interface presented in this research maintains concurrency between views as well as concurrency with the underlying model object graph. By doing this, the designer is able to perform functions in the interface that is most suitable for that operation. Any additions or changes to the system are reflected across all design views.

Alpha _1

The Alpha_1 system is a feature-based CAD research system developed at the University of Utah. Geometric data is maintained by an underlying model object graph. Alpha_1 incorporates a client-server distributed architecture. (Figure 2) This enables both a central representation for design consistency and distributed display for high user interface performance.

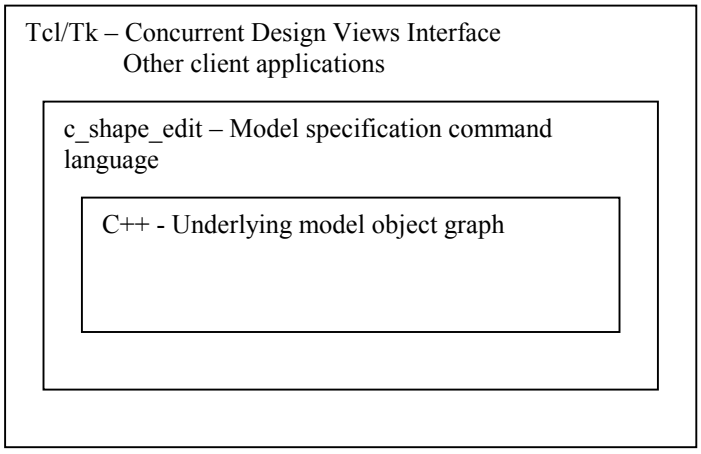

Figure 2. Client-server architecture of the Alpha_1 system

The concurrent design views interface presented in this paper was written at the application level in Tcl/Tk. This enables the concurrent design views interface to communicate with the server application in the same manner as other client applications. In addition, the concurrent design views interface was specifically designed to augment existing client applications, thereby providing additional user interface capabilities and overall system power.

DESIGN RESULTS

The design of complex products can be greatly facilitated by employing the complexity management tactics described in previous sections of this work. As an example to illustrate the ability of the principles and design tools presented in this research to manage design complexity, the design of a formula automobile is presented in this section.

Incremental Design Example – Formula Automobile

Each year, as part of a national competition sponsored by the Society of Automotive Engineers (SAE), an undergraduate design class at the University of Utah designs, builds, and races a prototype formula automobile (FormulaSAE). Each automobile contains hundreds of components allocated in many different sub-assemblies. Therefore, the FormulaSAE automobile provides an illustration of the capabilities of the concurrent design views interface presented in this research.

Due to short development time constraints and high complexity, multiple design teams work simultaneously on different portions of the overall design. The formula automobile is initial divided into three main subsystems (body, chassis and power train). (Figure 3)

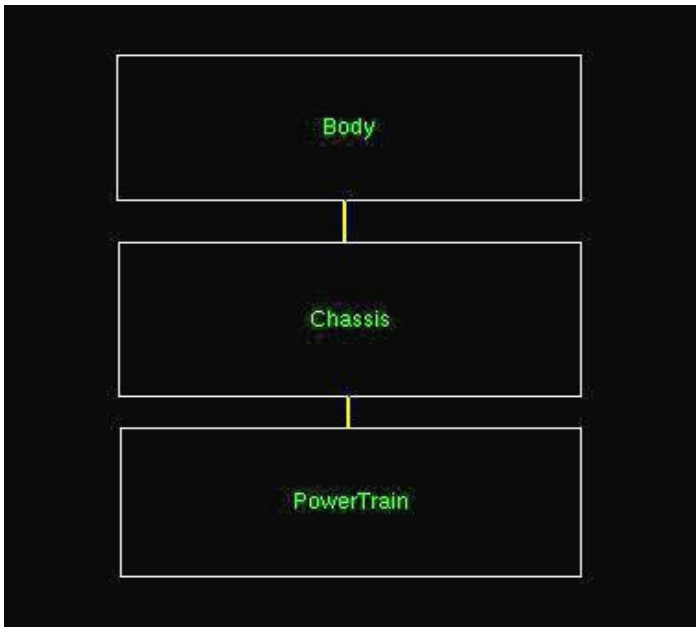

Figure 3. Initial decomposition of FormulaSAE automobile

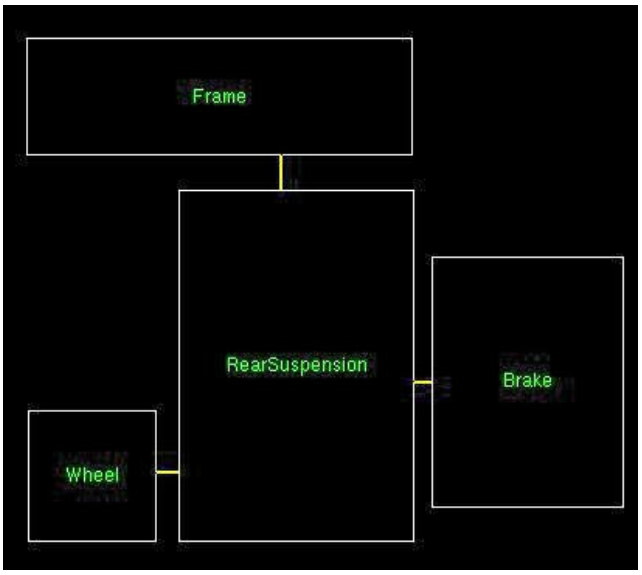

Figure 4. Decomposition of chassis sub-assembly

The chassis and power train portions of this design are still too complex for one design team to completely develop, so further decomposition of these subsystems is performed. To simplify presentation and repetition of similar practices, this example will concentrate on the chassis portion of the automobile.

The chassis is further decomposed into frame, wheel, rear suspension and brake sub-assemblies. (Figure 4) Each of these subassemblies as well as their relationships to other subassemblies can easily be created and understood in the concurrent design views interface by drawing the boxes and lines visible in each of the figures in this section.

The rear suspension and brake sub-assemblies are further decomposed into another lower level of abstraction to yield the final conceptual design of the chassis portion of the FormulaSAE automobile which is presented in Figure 5.

As stated earlier in this work, while creating the conceptual design of the formula automobile, a usable design framework is also created in the Alpha_1 environment. This design framework is represented by a structure of nested aggregation and relationship objects. Figure 6 is an example of the high level specification for the chassis sub-assembly of the FormulaSAE automobile that was created while generating the diagrams in Figures 3-5.

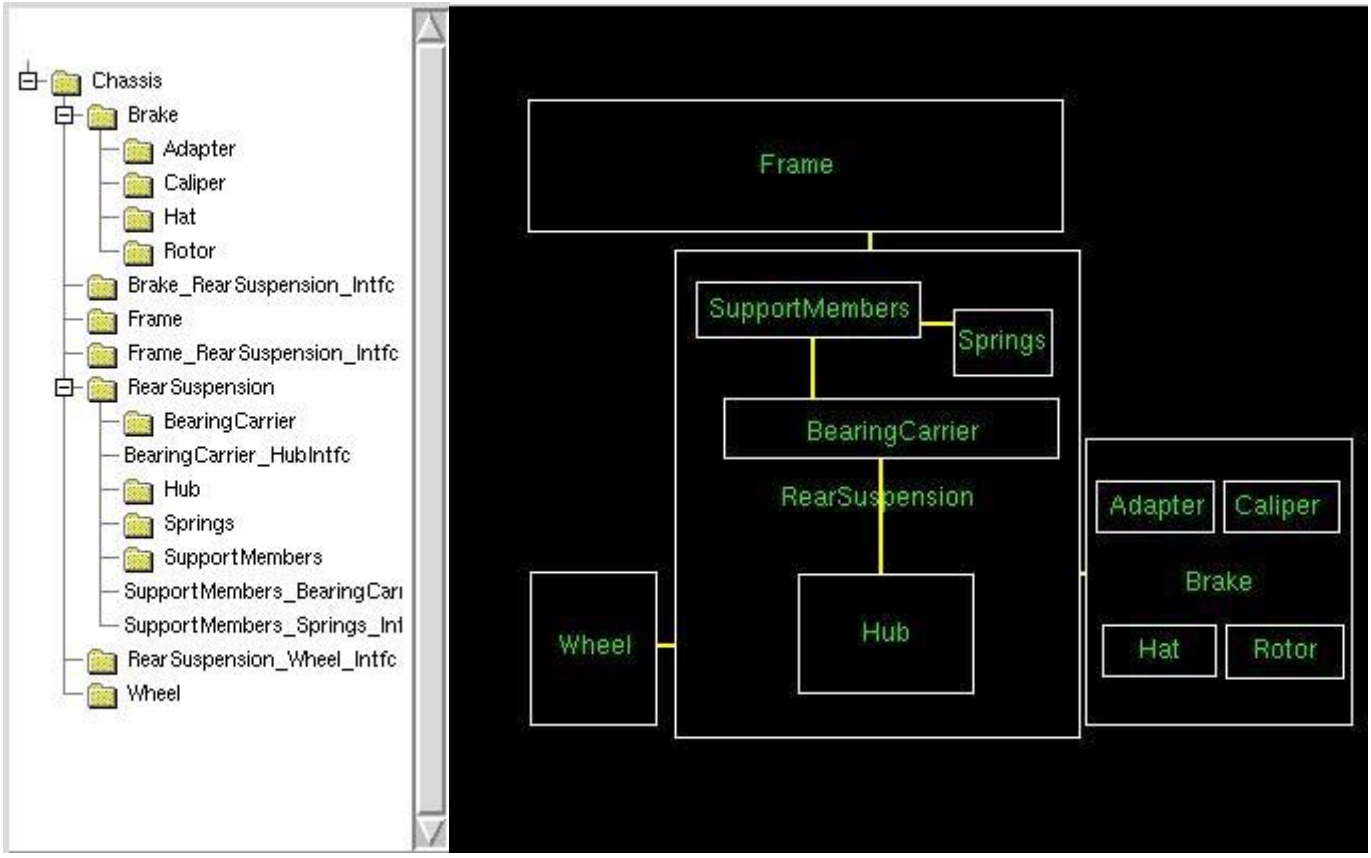

Figure 5. Decomposition of chassis subsystem of FormulaSAE automobile

```
Chassis : {
    Frame;
    Wheel;
    RearSuspension : {
        Support Members;
        Springs;
        BearingCarrier;
        Hub;
        SupportMembers_Springs_Intfc;

    SupportMembers_BearingCarrier_Intfc;
        BearingCarrier_Hub_Intfc;
    }
    Brake : {
        Hat;
        Rotor;
        Caliper;
        Adapter;
        Hat_Rotor_Intfc;
        Rotor_Caliper_Intfc;
        Caliper_Adapter_Intfc;
    }
    Frame_RearSuspension_Intfc;
    RearSuspension_Wheel_Intfc;
    Brake_RearSuspension_Intfc;
}
```

Figure 6. Model specification command language generated for the FormulaSAE automobile chassis.

After the high level specification has been generated, work on creating more detailed geometry may then be performed by the designer. In addition, the high level design framework is conducive to allowing multiple designers to work on separate portions at the same time, while maintaining the inter-relationships between components. The hierarchical structure of the design framework also allows geometric specification at different levels of complexity, ultimately leading to a final geometric specification. This detailed specification is suitable for manufacture as with other 3D CAD systems. An example of the final geometric representation of a portion of the Formula SAE automobile is presented in Figure 7.

CONCLUSIONS AND FUTURE WORK

We have presented a design views interface built upon an existing complexity management framework. This interface is capable of effectively managing design complexity by providing multiple levels of design detail and complexity and hierarchical decomposition of complex problems in an intuitive user interface. Design alternatives and history are also supported, thus providing an easy to use interface that aids the designer in visualization and analysis at all stages of the product development cycle. Concurrency between design views is maintained, so that changes made in any portion of the interface will be reflected across all interfaces.

We have integrated this complexity management interface into *Alpha_1*, an existing design system, and implemented interface to be compatible with existing Alpha_1 applications. An example of the use of this system to develop a product specification was presented in the formula automobile example.

ACKNOWLEDGMENTS

This work has been supported in part by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219), and DARPA (F33615-96-C-5621). All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of sponsoring agencies.

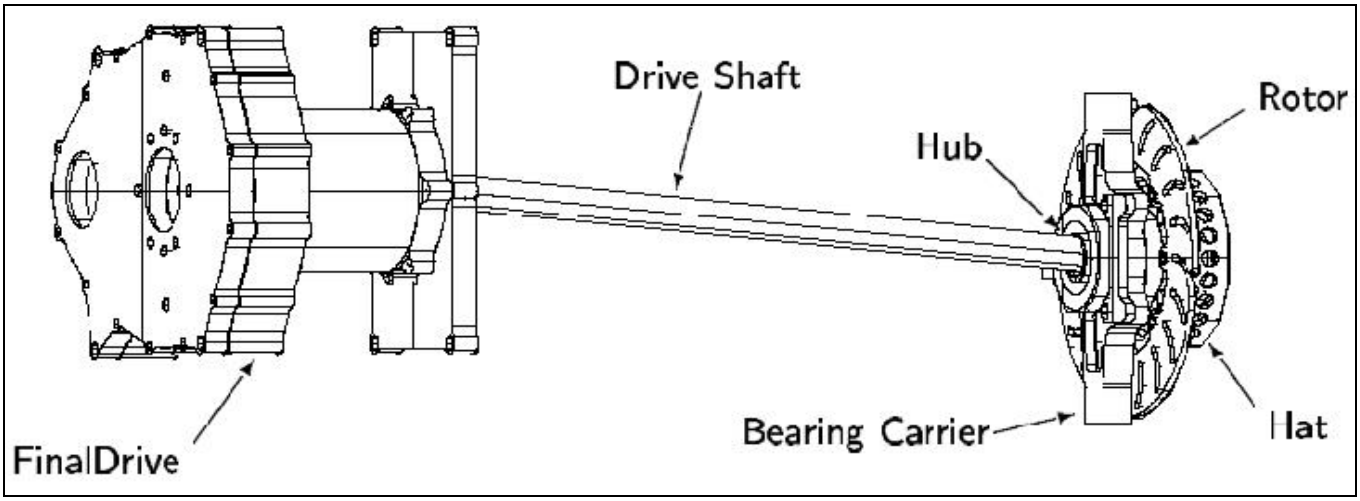

Figure 7. Detailed geometric representation of rear layout of the Formula SAE automobile

REFERENCES

C.-X. Feng, C.-C. Huang, A. Kusiak, and P.-G. Li Representation of functions and features in detail design. *Computer Aided Design 28* (12): 961-971, 1996.

S.R. Gorti and R.D. Sriram. From symbol to form: a framework for conceptual design. *Computer Aided Design 28* (11): 853-870, 1996.

T.M. Jacobs. *An automated framework for managing design complexity*. PhD Dissertation, University of Utah, 1998.

T.M. Jacobs and E. Cohen. Aggregation and Controlled Interaction: Automated Mechanisms for Managing Design Complexity. In *Proceedings of the 1998 ASME Design Engineering Technical Conferences.* American Society of Mechanical Engineers, 1998. DETC98/DTM-5677.

B. Loss. DRIVE: A system for interactive review of engineering designs. *University of Utah, Master's Thesis,* 1997.

M. Rosenman and J. Gero. Modeling multiple views of design objects in a collaborative CAD environment. *Computer Aided Design 28 (*3):193-205, 1996.

M. Saad and M.L. Maher. Shared understanding in computer-supported collaborative design. *Computer Aided Design 28* (3):183-192, 1996.

University of Utah, Department of Computer Science. *Alpha_1 User's Manual, Version 99.01,* 1999. Online HTML Document.

University of Utah, Department of Computer Science. *FormulaSAE*, 1997. Online HTML Document.

M. Zanella and P. Gubian. A conceptual model for design management. *Computer Aided Design 28* (1):33-49, 1996.