**Proceedings of DETC'98**
**1998 ASME Design Engineering Technical Conferences**
**September 13-16, 1998, Atlanta, Georgia, USA**

# DETC98/DTM-5677

# AGGREGATION AND CONTROLLED INTERACTION: AUTOMATED MECHANISMS FOR MANAGING DESIGN COMPLEXITY

**Timothy M. Jacobs and Elaine Cohen**
Department of Computer Science
University of Utah
50 South Central Campus Drive, Room 3190
Salt Lake City, Utah 84112-9205
Phone: (801) 581-5642 or 8235
Email: tjacobs@computer.org; cohen@cs.utah.edu

## ABSTRACT

Complexity in modern product design is manifest through large numbers of diverse parts, functions, and design disciplines that require an intricate web of synergistic relationships to link them together. It is extremely difficult for designers to assimilate or represent such complex designs in their totality.

In this research, we present a framework that utilizes the intricate relationships between design components to enhance the representational power of design models and to provide focal points for automating the management of design complexity. We introduce automated mechanisms, based on aggregation and interaction relationships between design components, that integrate model structure, a variety of conceptual and detailed design information, and product management controls into a single modeling framework. These mechanisms are easily incorporated into design models and they facilitate re-use and cooperative design by ensuring that related entities can be modified independently.

## INTRODUCTION

The complexity of modern product design manifests itself in many different ways. Typically, complexity results from a large number of parts in an assembly, complex geometry or multiple functions within an individual part, or the combination of many different design disciplines within a single assembly (Tegel, 1997). Quantity and diversity alone, however, do not completely capture design complexity. The individual components and functions are linked together in an intricate web of synergistic relationships through which the design becomes more powerful and complex than the sum of the individual pieces.

The intricate relationships and large quantities of information in a complex design are very difficult for a designer to assimilate. The relationships between design characteristics are often unknown or poorly understood which further exacerbates the problem. In an attempt to cope with these difficulties, a number of techniques have been developed for managing design complexity.

One technique is to break the design problem into a number of smaller sub-problems, each of which is less complex and more easily managed. If done properly, these smaller problems can be resolved simultaneously by separate design teams, then the solution can be integrated together to form the complete product design. In some cases, existing designs may be re-used to satisfy sub-problems.

In the early stages of design, complexity is frequently reduced by deferring specification and modeling of many of the details, both geometric and functional. High-level concepts are developed and analyzed to narrow down the set of possible solutions. Once a reasonable conceptual design is achieved, additional detail is added and the design is more rigorously analyzed. This cycle continues until the design has evolved into its final form.

As a design problem is decomposed into sub-problems or as detail is added at different levels of abstraction, additional relationships are established between components of the design. These relationships evolve along with the design components. Understanding and ensuring compatibility with these relationships is critical to designing a successful product. This is often complicated, however, by the difficulty in capturing and defining these relationships.

Ideally, one would dedicate sufficient resources to completely identify, specify, and analyze every aspect of a complex design. Since resources are often limited, however, one can reduce the chance of product failure by concentrating resources on those areas that pose the greatest risk. The relationships between design components have considerable impact on the overall design due to their synergistic effect. Consequently, these relationships provide a convenient focal point for minimizing design risk.

Computer-aided design systems tend to emphasize detailed modeling of individual design components and often fail to support the complex relationships between design components which are typical of most actual product designs. As a result, the design team must take additional steps to manage these relationships independent of the actual component models. Some CAD systems include product data management tools that can be used to specify and manage structural relationships between design components. These high-level tools, however, fail to capture complex design information, such as functionality, strength of materials, or geometric constraints, associated with these relationships. Alternatively, researchers have explored mechanisms for incorporating isolated details of the complex design relationships into the actual design model.

The extensible framework presented in this paper for managing design complexity brings together relationships between design components, detailed constraints and design information associated with these relationships, and methods for communicating and managing this information throughout the design model. We define an *aggregation* relationship which is used to specify the hierarchical structures which link together related components in support of design decomposition and abstraction at multiple levels of detail. *Controlled interaction* relationships are developed to describe how two parts work together to provide new capabilities. These relationships are embedded within the design model to provide focal points for representing, analyzing, and controlling the kinematic behavior, functionality, design goals, mechanical features, connectors and fasteners, complex geometry, component positioning, and other design information which contribute to design complexity.

To facilitate re-use and cooperative design, we ensure that design entities can be modified and validated independently. This is accomplished by incorporating aggregation and interaction information into the relationships between design entities, such that no modifications to the related entities are necessary. This allows the designer to retrieve a component from an electronic repository, validate the component against the interaction and aggregation relationships, and insert the component into the design model without having to modify the actual component. By not requiring modifications to existing design entities, it is also easier to integrate these mechanisms into existing design systems or to extend them to other design disciplines.

## BACKGROUND

Few computer-aided design tools provide support for managing complex design relationships. Instead, designers must document and track these relationships independent of the actual CAD model. This not only increases the burden on the design team, but it increases the risk of inconsistencies and errors during the design process.

Product data management tools have been integrated with some design systems to facilitate complexity management by allowing the designer to specify structural and classification relationships. These relationships, however, do not reflect the synergy which results from combining two or more components. Product management tools are limited to managing complete design files (drawings, specifications, analyses, etc.) and provide no modeling information such as functionality and constraints (Bilgiç and Rock, 1997). Due to the limitations of these management tools, considerable design and analysis must be accomplished outside of the tool. The results are then integrated back into the management tool as an additional step in the design process. This adds considerable overhead and limits the effectiveness of these tools for analyzing incremental design changes.

As an alternative to high-level data management tools, some researchers have incorporated fragments of information for managing complexity within the actual data model. These models, as summarized below, range from data structures which integrate structural and constraint relationships into the design model to mechanisms for simplifying the specification, within the design model, of the relationships which contribute to design complexity.

Eastman's *Engineering Data Model (EDM)* for architectural design is among the most comprehensive of these design models (1994; 1995). EDM provides aggregation, composition, and accumulation relationships which allow the designer to describe the aggregation hierarchy of the design along with constraint information between components. Due to its architectural focus on static structures, it is not clear that the relationships in EDM can incorporate mechanical interaction information such as forces, connectors, and kinematics.

Bordegoni and Cugini specifically address the interaction between fixed components in a mechanical assembly (1997). They propose an *assembly feature* for specifying the interaction relationship at various levels of detail. This is accomplished by having the designer fill in appropriate detail information in a cataloged template for each instance of an interaction relationship; however, if the template does not provide a slot for the information, the detail can be added only after modifying the template. Bordegoni and Cugini's assembly features have only been demonstrated for fixed assemblies with no kinematic interaction.

Gui and Mäntylä's *multi-graph* structure focuses on the top-down evolution of an assembly design from high level functional concepts (1994). The multi-graph supports multiple levels of detail and provides links between functional, structural, and geometric information. The multi-graph also provides a *connector* for describing force transmission and motion constraints associated with the interaction between parts in an assembly. The multi-graph emphasizes functionality and assembly modeling, but requires that the detailed manufacturing information be modeled separately. While linkages exist, the multi-graph mechanisms are not applied directly in the specification and validation of individual, manufacturable parts.

A number of researchers provide mechanisms to facilitate the specification of isolated pieces of the interaction and aggregation information. Lee's *mating features* are commonly used to describe the relative motion between parts (Lee and Gossard, 1985; Lee and Andrews, 1985; Kim and Lee, 1989; Beach and Anderson, 1996). Driskill proposes *assembly features* which are incorporated into each interacting part (1996), while Shah and Tadepalli present an *assembly feature*, separate from the two parts, which describes the interaction relationship (1992). Driskill's and Shah's assembly features identify compatible geometry and mating constraints that are useful for determining whether two parts may be assembled, but were not intended for specifying or controlling the interaction of the parts once assembled. Baxter uses an enhanced entity-relationship diagram to represent the functional relationships between parts (1994). Other research discusses mechanisms for representing and incorporating fasteners such as screws and rivets into the relationship between parts (Abrantes and Hill, 1996; Harper et al, 1997; Salomons et al, 1993).

## AUTOMATED MECHANISMS

This research distinguishes itself from other work in that it combines complex design relationships with detailed design and structural information to provide a powerful modeling framework for representing and managing complexity in mechanical design. We capture design information at different levels of detail, we support the hierarchical decomposition of design into multiple sub-problems, and we provide mechanisms for describing and controlling the synergistic relationships between design components.

We manage design complexity via automated mechanisms for specifying and controlling the aggregation and interaction relationships in a design. The *interface specification* is used for describing the many different aspects of the interaction relationship between two parts in an assembly such as relative motion, force transmission, geometry, and fasteners. Additional information, of any type, is incorporated into the interface or any other design component through the use of the *attachment* relationship. The attachment relation links two design components in a parent-child relationship and, along with the interface specification, defines the relationships necessary to form aggregations of design components which, in turn, are used to represent design decomposition and abstraction at multiple levels of detail.

### Aggregation

Developing and analyzing a design at multiple levels of detail is one approach to managing design complexity. The initial conceptual design may include a functional description and design goals with rough geometry, such as cylinders and boxes, or no geometry at all. Analysis of simple conceptual design models allows the designer to narrow down the set of possible design solutions without expending considerable resources. With this approach, additional detail is added and analyzed and the solution set is refined until a satisfactory detailed design is completed.

Similarly, a large assembly is often divided into sub-assemblies which are then designed independently of each other. When the design of an individual sub-assembly is completed, the sub-assembly is integrated into the next higher level of the assembly. Integration of these sub-assemblies would be simplified if each sub-assembly was designed to match the necessary interfaces with other sub-assemblies or parts.

We implement support for design decomposition and abstraction with *aggregation objects*, that is, objects used to collect related design entities. Aggregation objects are constructed from *interface specifications* (see next section) which specify how parts in an assembly should interact, and *attachment* relationships which define the parent-child hierarchy between design components at different levels of detail.

Although, the primary purpose of an attachment is to describe the hierarchical link between two design entities, the attachment also specifies the relative positioning of the two entities associated with the aggregation relationship. Other design information can be added as necessary.

We have defined three types of aggregations for grouping mechanical design information – *parts, assemblies*, and *neighborhoods*. Each of these may be modified and reconfigured to contain different components at each different level of detail.

```
SpindleHead : assembly {
    "Text describing design rationale,
     functionality, or other information
     may be included in an aggregation";

    "Parameters or goals may be specified";
    FatigueLife : 10000;
    Speed : 4000;

    "An assembly requires parts/sub-assemblies
     and interfaces to be specified";
    SpindleCartridge;
    HeadCasting;
    SpindleDrive;
    SpindleCartridge_SpindleDrive_Intfc :
        intfc( SpindleCartridge, SpindleDrive );
    SpindleCartridge_HeadCasting_Intfc :
        intfc( SpindleCartridge, HeadCasting );
    SpindleDrive_HeadCasting_Intfc :
        intfc( SpindleDrive, HeadCasting );
}
```

Figure 1. SPINDLE HEAD SUB-ASSEMBLY SPECIFICATION

A *part aggregation* consolidates related design features which occur at different levels of detail. Part aggregations examine the attachment relationships between entities to align and to validate design components which, when grouped together, represent a single functional part. Part aggregations may be nested within other part aggregations to facilitate the representation of various abstractions or multiple functions within a single part.

An *assembly aggregation* groups interacting parts or other sub-assemblies into a mechanical assembly. Assembly aggregations examine the interface specification links between components to align components and to validate the interface constraints.

A *neighborhood* is a generic aggregation which provides a mechanism for grouping design components which may not fit into the part or assembly aggregations. This is useful for integrating pre-defined design components at early stages of the design when the relationships between the entities are not yet well specified.

As shown in Figure 1, an aggregation is constructed with a *part, assembly*, or *neighborhood* command followed by the contents of the aggregation enclosed in brackets. To form a part or assembly, the designer must specify the hierarchical or interaction relationships along with the components. In addition, the designer may include parameterized constraints and design goals, design rationale and descriptions, or other design information. Each aggregation can be accessed and manipulated as an independent design module which encapsulates the embedded information.

### Controlled Interaction

The interaction between parts in an assembly represents considerable complexity and risk in a design. This interaction combines parts to provide additional capability that is not possible with independent parts. Functionality, forces, and motion are are all concentrated and transmitted through the interface between parts. In addition, considerable information is required to adequately describe this interaction. This includes:

- the relative motion between parts,
- fasteners to hold the parts in place,
- connectors to transmit forces between the parts,
- constraint parameters and equations,
- design goals and objectives,
- relations between different design disciplines,
- the relative position of the two parts, and
- complex geometry.

We use a single relationship object, the *interface specification*, for specifying and controlling the interaction between parts in an assembly. The interface specification provides a powerful mechanism for managing design complexity. The advantages of this mechanism include:

- Controlling design evolution and limiting the solution space by constraining certain aspects of the interacting parts.
- Providing a focal point for design analysis and management by isolating the interaction constraints.
- Minimizing design complexity and the effect of changes by localizing behavior to either side of the interface specification.
- Providing a design specification for independent work by different design teams on separate design components.
- Reducing design incompatibilities by ensuring interacting parts do not violate the specification.
- Controlling change propagation to affected parts via the interface specification.

The interface specification is used to capture all design information relevant to the interaction between parts. The interface specification is implemented to include assembly features and positioning constraints for each side of the interface, a joint describing the relative motion, and an aggregation hierarchy which allows the designer to attach additional information and levels of detail.

The assembly features describe compatible features for each of the two interacting parts. Unlike other applications of assembly features, we do not restrict assembly features to any particular design capability such as design for assembly or functional design. Instead, we allow any functional, manufacturing, or form feature, or any geometric object available in the design system to serve as an assembly feature. Through the use of aggregation objects, we support the evolution of assembly features at multiple levels of detail. The assembly features may be specified in advance of the two components for which the interaction relationship is being specified and then incorporated into these components via the aggregation mechanisms. Any subsequent changes or additional detail added to the assembly features will cause the related parts to be automatically regenerated to include the changes. If the changes lead to invalid design models, the remainder of the model remains unchanged, and the designer is notified.

Positioning constraints are specified with user defined anchors. Each feature, part, or aggregation in the design model has a base anchor which is aligned with the positioning anchor. This mechanism is used to automatically align features, parts, and sub-assemblies when part or assembly aggregations are created. A change in one positioning constraint is automatically propagated throughout the entire assembly. For example, if the positioning constraint for a hole feature is changed, this will cause the hole to be relocated along with any component linked to the hole. If a shaft has been inserted into the hole, the shaft will be relocated, as will any other components linked to the shaft.

Common joint types such as *revolute, prismatic, spherical*, and *against* are pre-defined and are instantiated by specifying the valid expanse of motion and the current relative position. Parts are moved by modifying the relative position. Any movement is checked automatically to ensure it falls within the constraints of the instantiated joint.

As additional detail is specified or becomes clear, it is added to the interface specification through the aggregation hierarchy. Any object defined in the design system may be incorporated into the aggregation hierarchy; however, some specialized aggregations are particularly applicable to the interaction between parts in an assembly:

- Nested interface specifications accommodate the decomposition of the interface as additional components are added to the interacting parts or sub-assemblies.
- Fasteners and connectors support detailed force analysis across the interface and the generation of manufacturing features such as threaded bolt holes. Connectors include application information, such as bearing life and rotational speed, and incorporate force analysis information for assorted patterns of connectors. Electronic catalogs of fasteners can be used to facilitate fastener selection and instantiation.
- Applied forces or force constraints provide the information necessary to carry out preliminary and detailed force analysis.
- Additional constraints and analysis information, including information for interpretation by other tools, may also be incorporated into the interface specification.

The interface specification provides the capability to describe many aspects of the interaction between parts in an assembly. It also provides a focal point for invoking other computational tools and using the results to analyze the design model. When specified prior to part design, the interface specification can be used during part design to ensure compatibility. When used with pre-defined parts, the interface specification can verify that the two parts are compatible.

The interaction between parts may be specified at different levels of detail. Early in the design, the relative motion between parts and high-level parametric constraints may be the only information of interest. Eventually detailed constraints, forces, nested interfaces, fasteners and connectors, or other interaction information is specified.

Through the interface specification and its associated aggregation relationships, parts in the assembly are moved, design parameters are analyzed and modified, and assemblies are re-configured at different levels of detail and decomposition. Since the interface specification object is manipulated like any other design object in the system, the designer does not need to learn any new procedures or tool interfaces. The interface specification provides a powerful focal point for describing and controlling the complex relationship between interacting parts in an assembly.

#### Data Structures

The interface specification, attachment, and aggregation objects work together to support complexity management for a design. This is depicted in Figure 2 which summarizes the aggregation and interaction data structures for a simplified version of a milling machine spindle. In this figure, design entities are depicted as boxes, *attachment* relationships are shown as diamonds, and a kinematic joint is represented by a hexagon. Aggregations are surrounded by dashed lines.

Observe that the *interface specification* relationship is shown as an aggregation because it contains multiple design entities – the joint, the positive and negative assembly features, and the attached detail features. The *hole* and *shaft* features are included in both the part aggregations and the interface aggregation. This demonstrates how the interface specification can be incorporated into the actual design of the interacting parts to ensure compatibility.

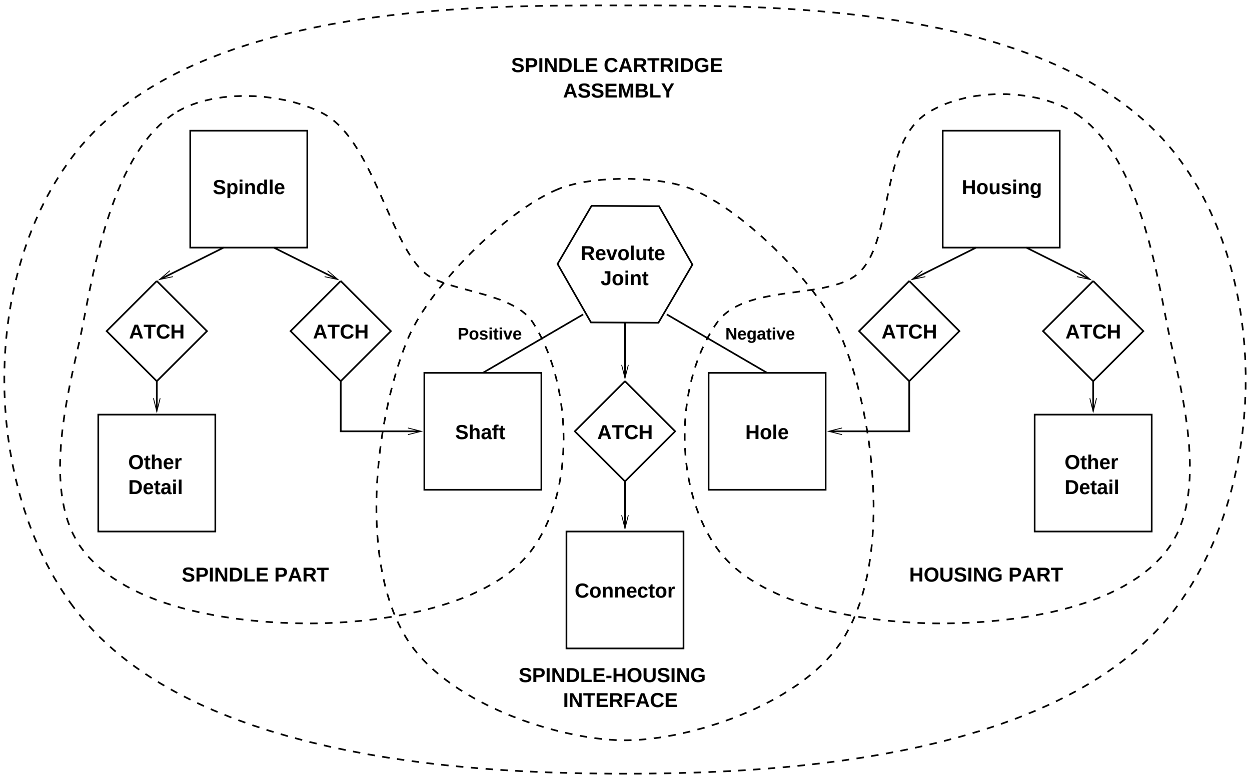



Figure 2. AGGREGATIONS AND RELATIONSHIPS

### Alpha_1

Our controlled interaction and aggregation mechanisms are integrated into *Alpha_1*, an object-oriented testbed system supporting research into geometric modeling, high-quality graphics, curve and surface representations and algorithms, engineering design, analysis, visualization, process planning, and computer-integrated manufacturing (University of Utah, 1995). *Alpha_1* provides geometric primitives, surface and curve representations, and mechanical features that can be used with the aggregation and controlled interaction mechanisms presented here to provide a powerful computer-aided design and manufacturing environment.

Models in *Alpha_1* are represented by a directed graph which identifies the prerequisite objects necessary to construct a particular object and the dependent objects that are based on the object. The model graph is used to propagate changes to dependent objects and to minimize processing by computing only the necessary prerequisite objects.

The *Alpha_1* object-oriented software development environment facilitates code generation for new modeling objects and provides a standard framework for building model object constructors to integrate model objects into graphical and textual user interfaces. The controlled interaction and aggregation mechanisms are implemented as independent *Alpha_1* model objects which can be manipulated and controlled like any other model object in the system.

## DESIGN RESULTS

The controlled interaction and aggregation mechanisms presented in this paper have been implemented and used in the design of a number of assemblies. Using an incremental design approach, a milling machine example is demonstrated here.

### Incremental Design

With their ability to handle decomposition and abstraction along with the specification of interaction constraints at multiple levels of detail, the mechanisms presented in this paper support a variety of design methodologies and analysis techniques. To fully demonstrate these capabilities, however, we present a design process in which the design evolves incrementally. In this process, the results are analyzed and validated after each design increment and the best approach for the next design increment is determined.

1. Specify external interfaces
2. Decompose into sub-assemblies or components
3. Specify internal interfaces
4. Design sub-assemblies or components
5. Analyze
6. If satisfied, then quit
7. Otherwise, refine in one of the following ways:
    - 7.1. Modify at same abstraction level
        - 7.1.1. Modify existing sub-assemblies, components, or interfaces
        - 7.1.2. Go to 5
    - 7.2. Add at same abstraction level
        - 7.2.1. Add additional sub-assemblies or components
        - 7.2.2. Go to 3
    - 7.3. Decompose at lower level of abstraction
        - 7.3.1. Map internal interfaces to external interfaces, as necessary
        - 7.3.2. Go to 2

Figure 3. INCREMENTAL DESIGN PROCESS

A pseudo-algorithm for this process is shown in Figure 3. This incremental approach supports controlled evolution of the synergistic complexity relationships while minimizing the chance that invalid design models will occur.

In a typical incremental design scenario, the operating environment for a product is identified and the environmental constraints and design goals imposed on the product design are described in interface specifications to components external to this design process. The designer then determines the major functions or concepts in the design and these major concepts become the initial design components. The potential interactions between the components are examined at a high level and the necessary interactions are specified as internal interface specifications. Some high level design information is incorporated into the components and the resulting design is analyzed to determine its feasibility and its ability to satisfy design goals. At this time the design team also plans the next increment of the design.

The design can be refined in a number of ways. If analysis reveals discrepancies in the design, these may be corrected by modifying some parameters or constraints. Alternatively, additional components may be added, at the same level of detail, to satisfy missing functionality. Once a satisfactory design is obtained at one level of detail, it may be decomposed further by adding additional detail and constraints.

At any detail level, independent design components or sub-assemblies may be designed simultaneously by different design teams. If similar design components already exist, these should be modified and re-used where possible.

```
MillingMachine : assembly {
    "Components in the milling machine assembly";
    VertDrive;
    Column;
    Bed;
    XYDrive;
    Table;
    SpindleHead;

    "Interfaces in the milling machine assembly";
    ToolHolder_SpindleHead_Intfc :
        intfc( ToolHolder, SpindleHead );
    SpindleHead_VertDrive_Intfc :
        intfc( SpindleHead, VertDrive );
    VertDrive_Column_Intfc :
        intfc( VertDrive, Column );
    Column_Bed_Intfc : intfc( Column, Bed );
    Bed_XYDrive_Intfc : intfc( Bed, XYDrive );
    XYDrive_Table : intfc( XYDrive, Table );
    Table_Fixture : intfc( Table, Fixture );
};
```

Figure 4. MILLING MACHINE DECOMPOSITION

### Milling Machine

A milling machine creates a manufactured part by cutting away excess material from a pre-formed piece of stock. Since designs for milling machines already exist, it is not necessary to completely design the machine. Instead, an existing machine design is decomposed into smaller sections which are more easily understood and analyzed. We then focus on the design of a single sub-assembly within the milling machine.

The milling machine is initially divided into six subassemblies: a spindle head for mounting and spinning the cutting tools, a drive for moving the spindle head in a vertical direction, a column for mounting the vertical drive, a table for mounting the work piece, an X-Y drive for moving the workpiece horizontally, and a bed upon which the column and the X-Y drive are mounted (University of Utah, 1997). This decomposition is specified as an *assembly* aggregation which includes the six milling machine subassemblies linked through interface specifications as shown in Figure 4a. The **`toolholder_spindle_intfc`** and the **`table_fixture_intfc`** are also added to the milling machine assembly to specify interfaces between external objects with which the milling machine must interact.

The spindle head is further decomposed into a spindle cartridge which holds and rotates the cutting tool, a spindle drive which provides the power to spin the spindle cartridge, and a head casting upon which the cartridge is

```
toolholder_spindle_intfc : intfc {
    joint : rigid();

    "Select geometry from standard tool holder";
    toolholder : toolholder_taper40;
    toolholder_slot : reverseObj( toolholder );

    "Identify and position positive and
     negative features of interface";
    pos : intfcpos( baseAnchor, toolholder );
    neg : intfcneg( basAnchor, toolholder_slot );

    "Attach forces acting on tool holder";
    atch1 : partof(joint, baseAnchor, axialForce);
    atch2 : partof(joint, XAnchor, radialForce);
};
```

Figure 5. TOOL HOLDER - SPINDLE INTERFACE SPECIFICATION

mounted. The spindle head decomposition is specified in a manner similar to the milling machine assembly and the milling machine assembly is automatically updated with the additional detail from the spindle head sub-assembly. To simplify the presentation, only the spindle cartridge design from the spindle head sub-assembly is demonstrated in this example.

The first step of the incremental design process is to identify the operating environment and the external constraints imposed on the design. The spindle cartridge design is constrained by the size of the tools, the required milling accuracy, and the desired cutting speed. Although not part of the spindle cartridge, the tool holder holds cutting tools that interact with the part being milled, thereby exerting forces on the spindle cartridge sub-assembly. These forces, along with the interaction of the spindle cartridge with the head casting and the spindle drive, must all be considered in the design of the spindle cartridge. To accommodate these external constraints, interfaces are specified from the spindle cartridge to the tool holder, head casting, and the spindle drive. Initially, we only want to analyze the relative motion and forces acting on the spindle cartridge, so these constraints are added to the interface specifications along with assembly features identifying the known geometry and manufacturing features associated with the interaction. As shown in Figure 5, the interface between the tool holder and the spindle is specified with a fixed joint and positive and negative features to accommodate the tool holder part. The interface object is similar to the part aggregation which allows the external forces acting on the tool holder to be attached with *partof* relations.

For the second step of the incremental design process,

```
SpindleCartridge : assembly{
    "Performance parameters and goals";
    FatigueLife : 10000;
    Speed : 4000;

    "Cartridge components";
    Spindle;
    Housing;
    Drawbar;

    "Interfaces between component";
    Spindle_Housing_Intfc :
        intfc( Spindle, Housing );
    Spindle_Drawbar_Intfc :
        intfc( Spindle, Drawbar );

    "External interface";
    Spindle_ToolHolder_Intfc :
        intfc( Spindle, ToolHolder );
};
```

Figure 6. SPECIFICATION OF SPINDLE CARTRIDGE SUB-ASSEMBLY

the spindle cartridge is decomposed into its major components as specified in Figure 6. We identify three major functional components: a spindle which rotates at a high rate of speed, a housing to provide a stable mounting for the spindle, and a draw bar for mounting the tool holder. As the spindle cartridge is decomposed, interactions are identified between the spindle, the housing, and the draw bar and incorporated into the sub-assembly with interface specifications. Design goals such as desired cutting speed and fatigue life are specified as parameters in the spindle cartridge sub-assembly.

To ensure compatibility between parts, essential joint and feature information is incorporated into the interface specifications (Step 3 of the design process). The spindle is a shaft which spins within the housing so we specify an interface containing a revolute joint with 360 degrees of rotation, a through-hole feature, and a cylindrical shaft feature (Figure 7). To be able to analyze the forces acting on this interface, we attach a force constraint to the interface specification to represent the estimated forces the interface should withstand. The draw bar shaft moves in and out of the spindle like a piston. For this interface we specify a prismatic joint with limited movement allowed along the axis, a hole feature, and a cylindrical shaft feature.

The design process proceeds by describing the high level design of the major components within the spindle cartridge sub-assembly (Step 4). To model the spindle part, the negative feature of the tool holder-spindle interface and the hole

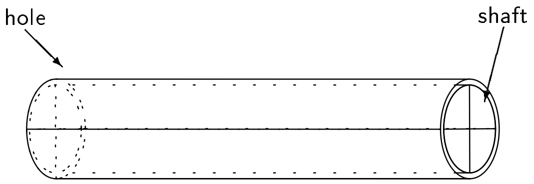
```
spindle_housing_intfc :* intfc {
    joint : revolute( 360 );

    "Positive feature is a shaft";
    profile : profile( ... );
    shaft : surfrev( profile, true );
    pos :* intfcpos( baseAnchor, shaft );

    "Negative feature is a hole";
    hole : entity( hole( ... ) );
    neg :* intfcneg( baseAnchor, hole );

    "Force constraint connector";
    forces : forceConn( 1000, vector( 1, 0, 1 ) );
    atch1 : partof( joint, baseAnchor, forces );
};
```

**(a) Interface Specification**




**(b) Feature Geometry**

Figure 7. INITIAL SPINDLE-HOUSING INTERFACE

```
spindle : part {
    "Extract features from interfaces";
    shaft : spindle_housing_intfc.posEntity;
    hole : spindle_drawbar_intfc.negEntity;
    holder : toolholder_spindle_intfc.negEntity;

    "Make features partof spindle";
    atch1 : partof(shaft, holderPosition, holder);
    atch2 : partof(shaft, drawbarPosition, hole);
};
```

Figure 8. SPINDLE PART SPECIFICATION

feature of the spindle-drawbar interface are attached with *partof* relations to the shaft feature of the spindle-housing interface as shown in Figure 8. The spindle shaft and its attachments are specified as a *part* aggregation which automatically aligns the attached parts and incorporates them into a part model for the spindle. The housing and the drawbar parts are completed in a similar fashion. By deriving parts from the interface specifications, any changes in the interfaces will be automatically reflected in the part

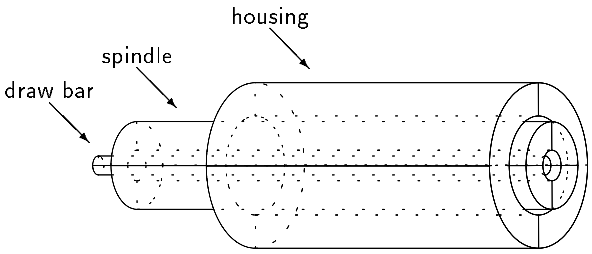



Figure 9. INITIAL SPINDLE CARTRIDGE SUB-ASSEMBLY

model. This also ensures that the parts are compatible with the interfaces.

As the interfaces and parts are specified, the spindle cartridge sub-assembly is automatically updated. The assembly aggregation uses the spindle part as a base part and automatically aligns the remaining parts according to the positions in the interface specifications. Figure 9 shows the resulting geometry of the initial spindle cartridge sub-assembly. The associated aggregation, attachment, and interface structures for the spindle and housing assembly were shown previously in Figure 2.

Now that the initial design increment is completed, analysis is performed to validate the design, evaluate its satisfaction of design goals, and determine how to proceed with the next increment (Step 5 of design process). With just a rough description of the part geometry and the interfaces, we are able to perform preliminary analysis to determine if the design approach is reasonable. We invoke automated mechanisms, which focus on the specified aggregation and interaction relationships, to analyze the forces acting on the entire assembly. Positioning of joints is automatically validated as the assembly is updated. The designer may manipulate joint positions or key parameters upon which the joints are dependent to analyze the kinematic behavior of the sub-assembly. If problems are discovered we focus the analysis on individual interfaces to isolate the problems. Constraints, parameters, and design components are refined as necessary (Step 7.1 of design process), and the component models automatically regenerated, until all problems are resolved.

When we are satisfied with the results of the first design increment, we refine the design by adding additional detail (Step 7.3 of design process). One of the key components in determining milling speed and fatigue life are the bearings between the spindle and the housing. To incorporate these bearings into the design, parametric models of bearings with the proper dimension and estimated force capacity are automatically retrieved from an electronic catalog. Alternatively, the designer can specify the desired bearing

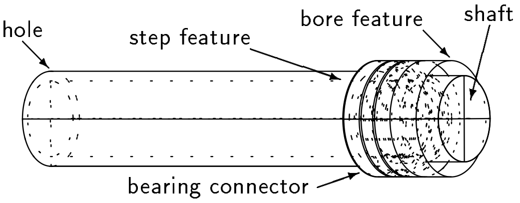
```
spindle_housing_intfc :* merge {
    "Retrieve bearing from catalog"
    bearing : lookupbearing( "L13", bearingCat );

    "Insert 2 bearings into connector,
     specify fatigue life and speed goals,
     and attach to interface aggregation";
    connector : bearingconn(array(bearing, spacer,
                                  invert(bearing)),
                              FatigueLife, Speed );
    atch1 : partof( joint, connAnch, connector );

    "Generate step features and make associated
     shaft part the new positive feature";
    shaft_part : part{
      shaft : shaft;
      shaftstep : bearingConnStep( .... );
      atch1 : partof(shaft, stepAnch, shaftstep);
    }
    pos :* intfcpos( baseAnchor, shaft_part );

    "Generate bore features and make associated
     housinghole part the new negative feature";
    hole_part : part{
      housinghole : hole;
      bore : bearingConnHole( ... );
      atch1 : partof(housinghole, boreAnch, bore);
    }
    neg :* intfcneg( baseAnchor, hole_part );
};
```

**(a) Interface Specification**




**(b) Feature Geometry**

Figure 10. DETAILS OF SPINDLE-HOUSING INTERFACE

parameters in a new model object. We insert the bearings into a bearing connector in which we also include the design goal parameters specifying desired fatigue life and rotational speed. The *merge* command incorporates the connector into the spindle-housing interface and attaches the connector to the original joint as specified in Figure 10a. Bearing features are generated from the connector and attached to the positive and negative interface features. The interface geometry is automatically updated as shown in Figure 10b. The housing and spindle parts are regenerated

```
housingAssy : assembly {
    housing;
    nosecap;
    nosecap_housing_intfc :
        intfc( housing, nosecap );
};
```

Figure 11. HOUSING SUB-ASSEMBLY SPECIFICATION

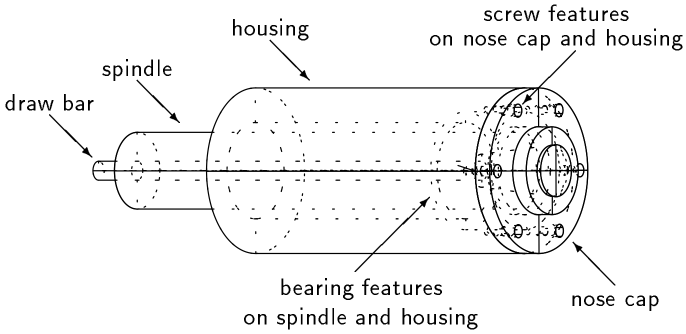



Figure 12. DETAILED SPINDLE CARTRIDGE SUB-ASSEMBLY

with the appropriate bearing connection features and these changes are automatically propagated to the entire spindle cartridge sub-assembly.

So that the bearings can be inserted during the assembly process and held in place during operation, the housing part is decomposed into another sub-assembly containing the main housing part and a detachable nose cap at the end of the housing (Figure 11). This requires a new interface between the nose cap and the housing. The nose cap is to be held in place with screws which are incorporated into the interface using a screw connector. The screws are retrieved from an electronic catalog and inserted in the screw connector which arranges six identical screws in a radial pattern. The screw connector is attached to the nosecap-housing interface, screw features are generated for the positive and negative interface features, and a new interface aggregation is created. By attaching the new interface to the housing part and specifying a new part aggregation, a model of the nose cap part is generated. This also automatically updates the model of the housing part to include threaded hole features for the screw connector. After updating the individual parts and interfaces, the model of the spindle cartridge sub-assembly is regenerated with the nosecap-housing interface and the new parts added (Figure 12).

With the completion of another design increment, we now need to confirm that we used the proper bearings and screws in our interfaces. Each connector has characteristics, such as bearing life or screw grade, which determine the force limits which the connector can withstand. The automated force analysis calculates these limits and determines if the applied forces exceed the capabilities of the connectors.

At this point in the design process, we have encapsulated descriptions of functionality and design rationale; design parameters, constraints, and goals; forces and kinematic information; manufacturing features; and geometry into the interface specifications of the spindle cartridge sub-assembly. From this information analysis can be accomplished for the forces and kinematic behavior of the sub-assembly, geometric interference can be calculated , or manufacturing process plans can be generated. We can generate different alternatives for the components or interfaces and automatically reflect changes in the affected parts and sub-assemblies. We have also decomposed the milling machine design problem into small sub-assemblies which are more easily understood and managed, and we have restricted the design of the spindle cartridge sub-assembly through external interface specifications which ensure its compatibility with the remainder of the milling machine.

## USABILITY AND EXTENSIBILITY

One of the objectives of this research is to provide automated support for complexity management as a natural part of the design process. This means commands for managing complexity must be similar to other design commands and must not require a significant effort on the part of the designer. To fit naturally into the design process, it is our hypothesis that complexity management must be implemented in fundamental design objects within the automated system. These object data structures must also be extensible to support a wide variety of design applications.

Analysis of the results presented earlier in this paper supports the premise that fundamental design objects can be useful for managing complexity. Controlled interaction and aggregation are accessible in the same fashion as any other design object in the design system (*Alpha_1*) and are completely integrated to work with the other design objects. Commands for complexity management are invoked the same as those for curves, surfaces, and other design objects.

While this research has not concentrated on user interface issues, the low-level interaction and aggregation structures have been incorporated into existing *Alpha_1* textual and graphical user interfaces. Since these data structures were implemented as standard design objects, embedding them in the user interfaces requires the same standardized programming steps as needed for other design objects.

Using design objects also proved to be extremely efficient for extending the complexity management capabilities to different mechanical applications. Once the primary aggregation and interaction structures were in place, it was a simple, straightforward task to add new connectors, constraint analysis, and management information. One example is the addition of the screw connector. A design object with the necessary parameters was defined and, using *Alpha_1*'s development environment, most of the code required for integration with the rest of the system was generated automatically. The only manually generated code was that necessary for the screw geometry and the force capacity calculations.

## SUMMARY AND CONCLUSIONS

In this research, we utilize the intricate relationships between design components to capture and control the synergistic effect these relationships have on design complexity. We combine these relationships with product management controls and design information at multiple levels of detail to provide a single framework for managing complexity in the design of mechanical assemblies. Model structure and organization are described in aggregation relationships which are used to manage design decomposition and abstraction. The functional, kinematic, geometric, and other information relevant to the interaction relationships between parts is incorporated into an interface specification which describes how two or more components are combined to produce additional functionality and complexity. Using the aggregation and interaction relationships as focal points, we provide automated mechanisms for analyzing the design model from multiple perspectives.

In addition to describing the interaction between parts, the interface specification can be used to validate compatibility between parts. If specified in advance of the parts, the interface specification can be incorporated into the actual design of the parts to ensure compatibility. Specifying the interface in advance facilitates design modifications since changes can be made at the interface between the parts and propagated to all affected parts. The work required of the designer is reduced since the detail is specified once in the interface rather than in each part. The interface provides a specification for simultaneous design of interacting components and facilitates integration of completed components into the higher level design model.

We have integrated this complexity management framework into *Alpha_1*, an existing design system, and implemented the complexity management objects so that they are specified and manipulated in the same fashion as other design objects in *Alpha_1* and require little additional learning curve. The mechanisms presented in this paper have been demonstrated in the design of a number of assemblies, including a milling machine and a student built automobile.

By implementing complexity management as relationships between design entities, no modifications are required to existing design components. This facilitates re-use of existing designs, integration of these mechanisms into existing design systems, and extension of the framework to include other design features or disciplines.

The mechanisms presented in this paper provide a framework for managing design complexity. By incorporating additional constraints, connectors, features, and other information, the framework can be extended to manage other design parameters or different design disciplines. A well-defined data structure also facilitates standardization and communication between different design applications. Appropriate communications and user interface capabilities will further simplify the use of these mechanisms and extend their utility to cooperative work across multiple terminals or across the Internet.

## ACKNOWLEDGMENT

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 the sponsoring agencies. Thanks also go to the students and staff of the *Alpha_1* project, within which this work was developed.

## REFERENCES

M. J. Abrantes and S. D. Hill. Computer-aided planning of mechanical assembly sequences. Technical Report 95-7, Monash University, Clayton, Australia, Feb 1996.

J. E. Baxter, N. P. Juster, and A. de Pennington. A functional data model for assemblies used to verify product design specifications. *Proceedings of the Institution of Mechanical Engineers*, 208(B4):235–244, 1994.

D. Beach and D. Anderson. A computer environment for realistic assembly design. In *Proceedings of the 1996 ASME Design Engineering Technical Conferences*. American Society of Mechanical Engineers, 1996. 96-DETC/CIE-1336.

T. Bilgiç and D. Rock. Product data management systems: State-of-the-art and the future. In *Proceedings of the 1997 ASME Design Engineering Technical Conferences*. American Society of Mechanical Engineers, 1997. DETC97/EIM-3720.

M. Bordegoni and U. Cugini. Feature-based assembly design: Concepts and design environment. In *Proceedings of the 1997 ASME Design Engineering Technical Conferences*. American Society of Mechanical Engineers, 1997. DETC97/CIE-4266.

E. E. Driskill. *Towards the Design, Analysis, and Illustration of Assemblies*. PhD thesis, University of Utah, 1996.

C. M. Eastman. A data model for design knowledge. *Automation in Construction*, 3:135–147, 1994.

C. M. Eastman, H. Assal, and T. Jeng. Structure of a product database supporting model evolution. *Proceedings of CIB Workshop on Computers and Information in Construction: Modeling of Buildings through their Life-cycle*, 1995. Stanford, California.

J.-K. Gui and M. Mäntylä. Functional understanding of assembly modelling. *Computer Aided Design*, 26(6):435–451, Jun 1994.

B. Harper, Z. Siddique, and D. Rosen. A computable fastener representation to support computer-aided configuration design for the life cycle. In *Proceedings of the 1997 ASME Design Engineering Technical Conferences*. American Society of Mechanical Engineers, 1997. DETC97/CIE-4310.

S. Kim and K. Lee. An assembly modelling system for dynamic and kinematic analysis. *Computer Aided Design*, 21(1):2–12, Jan/Feb 1989.

K. Lee and G. Andrews. Inference of the positions of components in an assembly: part 2. *Computer Aided Design*, 17(1):20–24, Jan/Feb 1985.

K. Lee and D. C. Gossard. A hierarchical data structure for representing assemblies: part 1. *Computer Aided Design*, 17(1):15–19, Jan/Feb 1985.

O. Salomons, J. Kappert, F. van Slooten, F. van Houten, and H. Kals. Computer support in the (re)design of mechanical products, a new approach in feature based design, focusing on the link with CAPP. *IFIP Transactions on Knowledge Based Hybrid Systems*, B-11:91–103, 1993. Electronic version from author's repository.

J. J. Shah and R. Tadepalli. Feature based assembly modeling. In *Proceedings of the 1992 ASME International Computers in Engineering Conference and Exhibition*, volume 1, pages 253–260. American Society of Mechanical Engineers, 1992.

O. Tegel. Support for handling complexity during product development. In *Proceedings of the 1997 ASME Design Engineering Technical Conferences*. American Society of Mechanical Engineers, 1997. DETC/EIM-3717.

University of Utah, Department of Computer Science. *Alpha_1 User's Manual, Version 95.06*, 1995. Online HTML Document.

University of Utah, Department of Computer Science. *Alpha_1 Spindle Project*, 1997. Online HTML Document.