Computer-Aided Design and Applications

http://www.cadanda.com

High-level Operations to Streamline Associative Computer-Aided Design

Nathan W. Scott[1] and C. Greg Jensen[2]

[1]Brigham Young University, intonate@gmail.com
[2]Brigham Young University, cjensen@byu.edu

ABSTRACT

Parametric, feature-based modeling with inter-part associativity allows complex assembly designs to be defined and re-defined while maintaining the vital part-to-part interface relationships. The top-down modeling method which uses assembly level control structures to drive child level geometry has proved valuable in maintaining these interfaces. This paper shows that programmatic operations can streamline this type of assembly and component-level design because a single programmatic operation can create an unlimited number of low-level features, modify geometry in multiple components, create new components, establish inter-part expressions, and define inter-part geometry links. Results from user testing show that a set of high-level programmatic operations can offer savings in time and effort of over 90% and can be general enough to support user-specified interface layouts and component cross sections while leaving the majority of the primary design decisions open to the engineer.

Keywords: associativity, design automation, assembly design.
DOI: 10.3722/cadaps.2009.317-327

1. INTRODUCTION

Well developed parametric CAD models can be reused to produce many similar designs, and can simplify the process of incorporating design changes [9]. In addition, associativity capabilities within modern CAD applications allow these parametric models to maintain relationships between features and components [12] which extends the benefits to models of entire assemblies. Creating robust parametric models is very time consuming, however, and "this level of skill can take years of training and experience to acquire" [2,4]. Parametric models of assemblies are especially dificult and tedius to model because of the many inter-part relationships that must remain associative. To increase modeling efficiency, modern CAD applications offer some high-level features for certain product types as well as User Defined Features (UDFs) which allow users to specify custom combinations of low-level features. These approaches to modeling assemblies are still restricted because they do not have any inter-part associativity or intelligence capabilities and can only combine a limited number of low-level features. They require users to create the inter-part expressions and geometry links by hand and do not allow the user to operate on multiple components at once or to create new components as part of the operation.

1.1 Objective

It is the objective of this research to show how the current modeling practices described above can be streamlined further through the use of product type-specific programmatic operations that would combine the benefits of UDFs and inter-part associativity and would function at a level much higher than inserting single features into individual components. Another objective is to demonstrate that

these operations leave the majority of the primary design decisions open to the engineer. This paper defines a programmatic operation as one that can create an unlimited number of low-level features, modify geometry in multiple components, create new components, establish inter-part expressions, and create inter-part geometry links. Since products of a similar type have similar features, programmatic operations can be written that can create most, if not all, of the CAD geometry necessary to define a product of a certain type.
This research will focus on the incorporation of design operations related to a specific class of products into an application that will interoperate with Siemens NX 5. The goal of this study is to develop a set of proof-of-concept applications that will streamline the design of assemblies and parts whose basic geometry has a uniform cross section (2 ½ dimensional). The specific test case that will be used to demonstrate and validate this research is the interstage sub-assembly on solid motor rockets. This particular assembly is a good candidate for this research because it is used repeatedly on a wide range of rockets and because its geometry is relatively simple.

1.2 Prior Approaches

A wide range of research has been performed in the field of computer aided design, especially in developing methods to streamline the product development process. Parametric, feature based modeling and relating parameters with equations are fundamental methods of reducing operations [4,7]. User-Defined Features (UDFs), which allow the user to define a set of standard, low level features that will be grouped together, have also been used with much success in such applications as aircraft skin lightening pockets and plastic snap features [1,4-6,9,10]. An especially effective method for maintaining associativity between parts in assemblies is the top-down modeling method which uses control structures in assembly level parts to maintain interfaces between its children level components [3,7,8].
The literature lacks a well developed method for automating the use of control structures to define custom assembly configurations, which will be the contribution of this paper. This paper also builds upon and extends the literature's methods of using high-level sets of features to minimize the effort required to define the model. However, the programmatic operations of this method are able to operate on a much higher level than inserting individual features into single components. Each operation can create large sets of features in multiple parts as well as the inter-part expressions and geometry links needed to maintain associativity.

2. METHODS

The methods discussed in this section represent a set of automated steps that can be used to design products in a certain geometric class. Although these methods are specifically chosen for their applicability to a specific class of products, they will be presented with as much generality as possible to enable their extension into other product classes. These methods were developed using the NX Open API in C++ and this paper uses language specific to that programming and modeling environment; but, the content of the methods should be general enough to apply to any similarly capable environment.

2.1 Overall Method for Design Automation

The steps used in automating the design of assemblies and components employ the control structure-based top down method. The steps can be grouped into three key operations.

> *Operation 1:* Define the control structure and part ownership and create the geometric links between the parents and children for the primary design features.
> *Operation 2:* Define the cross sections of children level parts, make them into solids, and add detail features such as chamfers, blend radii, and fastener holes.
> *Operation 3*: Add secondary design features. (This operation has been implemented for certain secondary features, but is outside the scope of this paper. See [10].)

For the remainder of this paper, *A* will represent an assembly-type part (a parent to at least one other part) and *C* will represent a component type part (one having no children). The superscript of *A* or *C* will represent the hierarchical level of the part and the subscript will represent its position relative to its sibling parts, e.g. A^2_1 is the first child of its parent and is a second level assembly, and C^3_2 is the

second child of its parent and is a third level component. No subscript will be used when referring to the collection of all parts on a certain hierarchical level. This is illustrated in Fig. 1(a).
In addition, let $\bigcirc_{Sk}(\ )$ represent an operation on sketch geometry. The notation from set theory for boundaries (bS) will be modified by a subscript 2 or 3 to distinguish between two-dimensional boundaries b_2 (sketch geometry) and three-dimensional boundaries b_3 (faces and edges of the solid). Fig. 1(b) illustrates these terms as well as other key terms that will be introduced in this section. b_2C^2 represents the two dimensional boundary of a component. b_3C^2 represents the three dimensional boundary of a component. I_{ij} is a unit of the control structure which represents the interface between components i and j. The clearance between components is denoted by ε and the chain link symbol denotes the sketch constraints between I_{ij} and b_2C^2. A^1 is the top level assembly that contains the control structure.

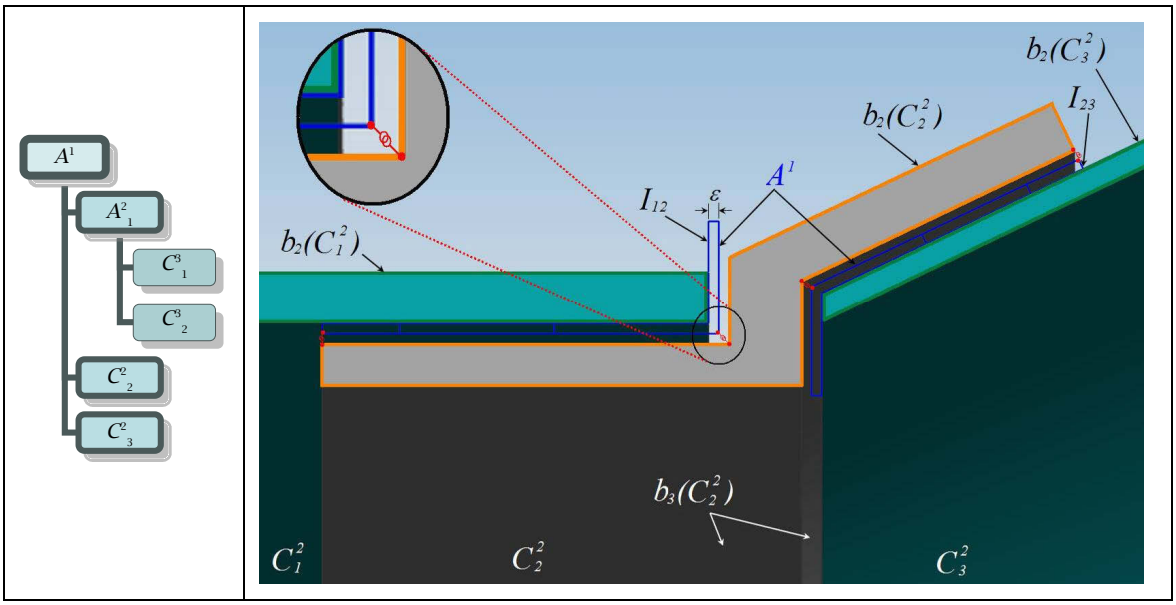

Fig. 1: (a) Assembly part notation (b) key terms.

2.2 Operation 1: Control Structure, Part Ownership, Geometric Links

The first operation helps the designer define the overall layout of the entire assembly and automatically creates the associative links from the top level control structure owned by A^1 to its children (A^2 and C^2). The control structure can be represented mathematically as the set of all points at which its children's boundaries (b_2C^2) are located within a given clearance of each other (ε). Let I_{ij} denote the intersection between components C^2_i and C^2_j.

$$I_{ij} = b_2C_i^2 \overset{\varepsilon}{\bigcap} b_2C_j^2 \tag{1}$$

Then A^1 is the sum of all intersections between its n children.

$$A^1 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} I_{ij} \tag{2}$$

See Fig. 1(b) for illustrations of I_{ij}, b_2C^2, ε, and A^1.
When modeling assemblies, a designer will often start with a hand drawing of the overall layout and will therefore know by inspection what the control structure needs to look like. The automated

method serves as a tool to quickly go from an arbitrary hand drawn assembly configuration to fully defined models with associativity. An example of an assembly layout can be seen in Fig. 2 (a) with the corresponding assembly control structure in Fig. 2 (b). While this layout of a rocket interstage assembly will be used to describe and evaluate the proposed methods, it should be noted that these methods are applicable to other design situations involving 2 ½ dimensional geometry.

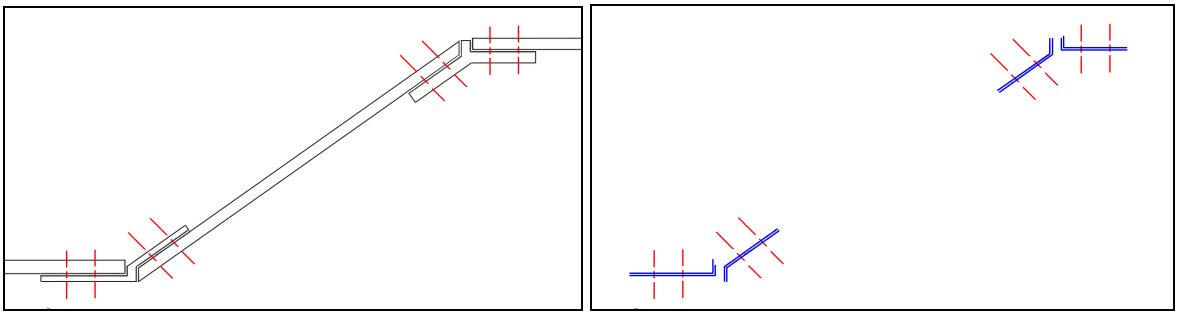

Fig. 2: (a) Sample assembly configuration, (b) assembly interfaces/control structure.

2.2.1 Application Architecture

The control structure is defined through a framework of C++ class objects (Fig. 3 (a)) which are instantiated based on user inputs from a graphical user interface (GUI) as illustrated in Fig. 3 (b). The Interface Object class contains all the data needed to define the geometry of one unit in the interface control structure (I_{ij}). It contains the member function $\bigcirc_{Sk}(I_{ij})$ to create the Sketch control geometry, and the member function $\overset{f}{\bigcirc}_{Sk}(I_{ij})$ to create the fastener control geometry based on the member variables. It also contains the member function $I_{ij}(C_i^2) = \bigcirc_{Wl}(I_{ij}(A^1) \rightarrow C_i^2)$ to make an associative copy of a unit (I_{ij}) of the assembly control structure in a child component C^2_i. This operation will be referred to as Wave Linking. The member function $\bigcirc_{Sk}(b_2 C_i^2 = I_{ij}(C_i^2))$ constrains the sketch geometry of the child part ($b_2C^2_i$) to its wave linked geometry ($I_{ij}(C^2_i)$). The Interface Derived Part class contains a collection of interface objects as well as its position on each interface. It also has a function that calls each of its interface objects' functions.

2.2.2 Application Procedure

The application is launched from within the CAD environment. First, a GUI collects from the user information such as what interfaces are in the assembly, what their dimensions and orientations are, what parts are in the assembly, and what interfaces are associated with each part (See Fig. 3(b)). After the data is collected from the GUI, the application automatically performs the following steps associated with operation 1.

1. Create part files for A^1, A^2 , and C^2.
2. Create control sketch in A^1
    a. Create and open sketch feature
    b. For each interface object: call MakeControlGeometry(Data)
    c. Close sketch feature
3. For each interface object: call MakeFastenerControlGeometry(Data)
4. For each Interface Derived Part Object
    d. For each Interface Object in collection: call LinkChildToParent
    e. Create and open sketch feature
    f. Call SketchMyInterfaces(MyInterfaceObjects, MyInterfacePositions)
    g. Close sketch feature

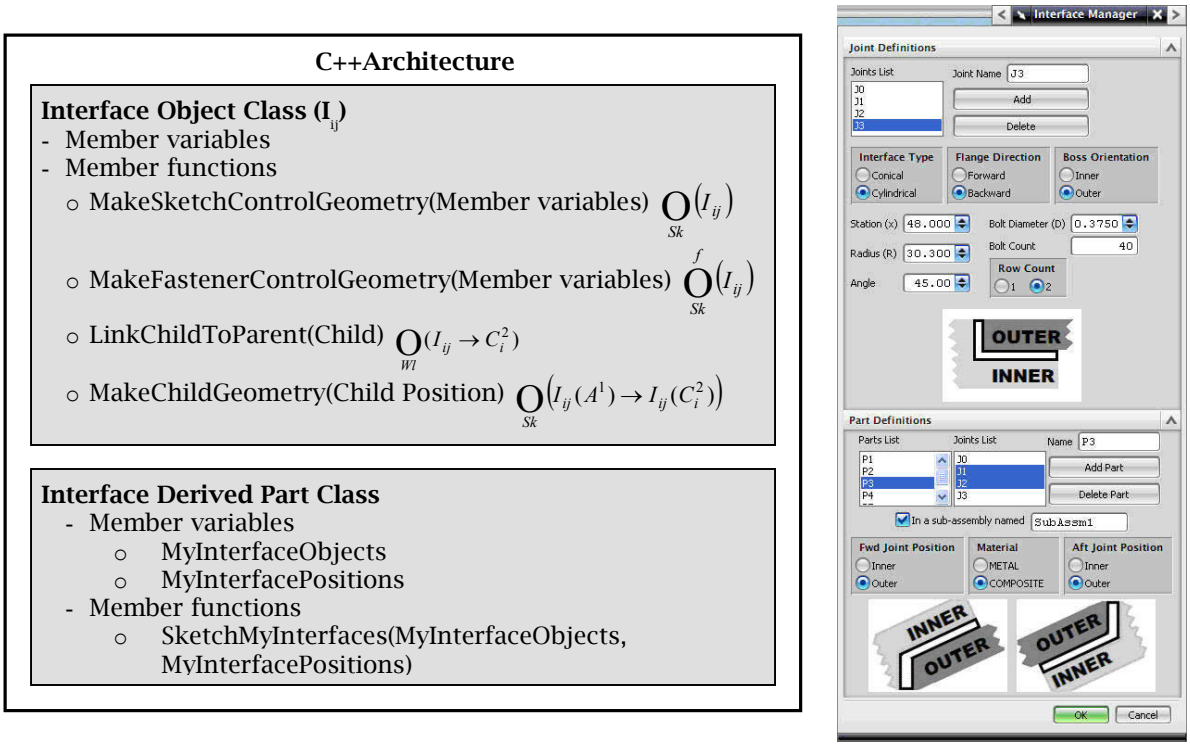

Fig. 3: (a) Class architecture and (b) the interface manager GUI.

At this point each part will have the geometric links associated with the assembly's control structure and will have component-specific sketch geometry that is constrained to the control structure. A sample part sketch would look like Fig. 4, where the blue lines represent the wave linked control structures $I_{12}(C_2^2)$ and $I_{23}(C_2^2)$, the orange lines represent the sketch geometry (a subset of $b_2C_2^2$), and the red points indicate constraints between the sketch geometry and the linked control structures which are shown as red chain links in Fig. 1(b). The component bodies are shown in gray for reference.

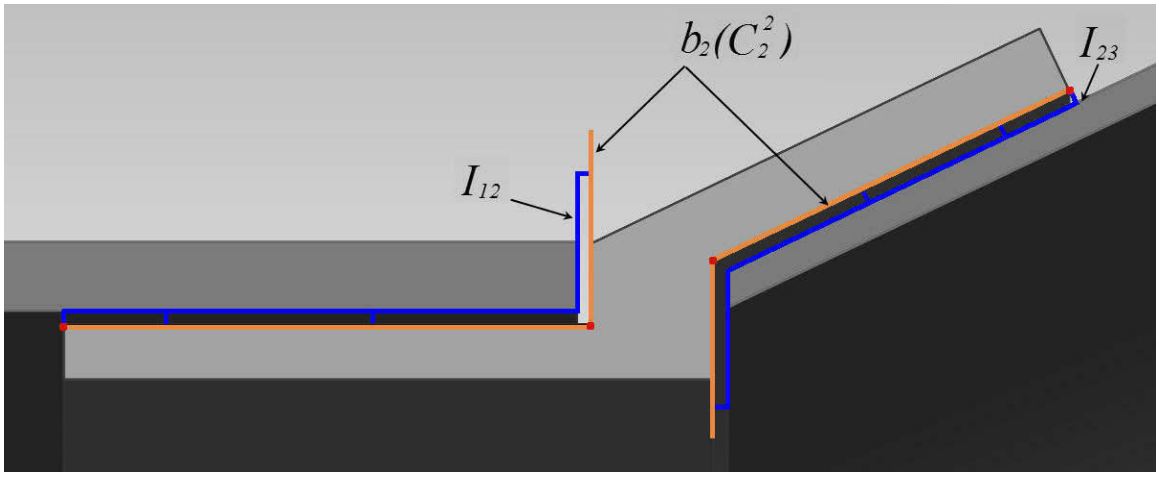

Fig. 4: Completion of operation 1 showing the sketch of one part.

2.2.3 Scope
This framework is general enough that any interface type can be created by defining a class that inherits from the Interface Object Class. Possible interface types that may be created using this framework include a manacle joint, a weldment, or riveted joints. For this paper, two interface types have been implemented that represent cylindrical and conical bolted flange joints.

2.3 Operation 2: Cross Sections, Solids, Detail Features
The primary steps associated with Operation 2 are

1. Complete the part cross section sketches
2. Create solids from sketches
3. Apply blends and chamfers to edges
4. Create fastener hole features and patterns

2.3.1 Step 1: Part Cross Sections
The first step of Operation 2, completing the part cross section sketches, is performed interactively by the designer. This is where the most variability exists in the design and since interactive sketching is fairly easy and fast, it is more advantageous to not hard code the creation of individual part sketches. The use of macros, or a library of applications to generate common sketches would accelerate this step, but will be left for future work. It should also be noted that if this step were automated Operation 2 could effectively be combined into Operation 1. This step completes the definition of the two-dimensional boundaries of the components (b_2C^2) which is illustrated in Fig. 5.

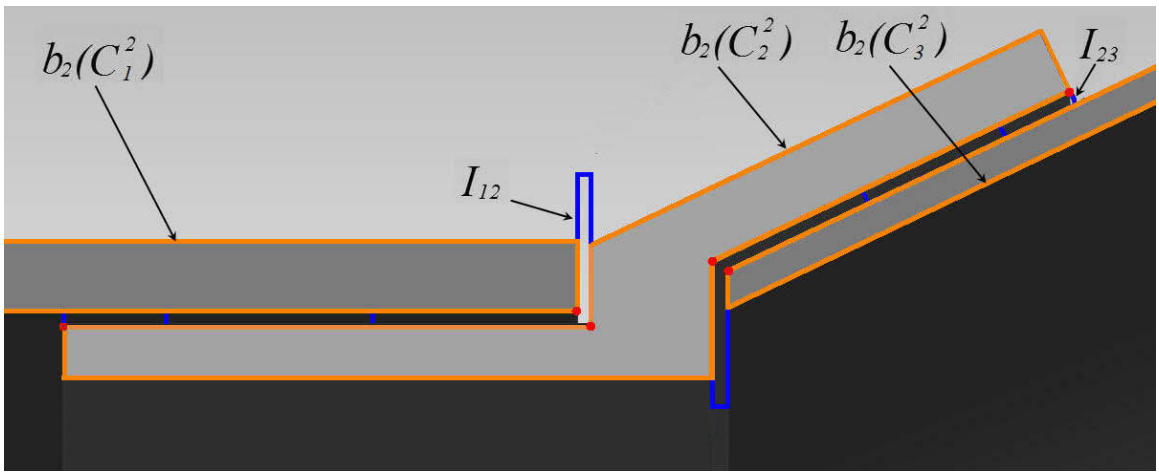

Fig. 5: Completion of operation 2 step 1.

2.3.2 Step 2: Solids
It is programmatically trivial to create the solidifying feature from the cross section sketches. As long as the sketch is named according to a pre-determined convention it can be retrieved and either extruded or revolved. Certain interface types may also contain parameters needed to perform this step such as an extrusion distance or revolve angle and would therefore create the necessary variables as part of Operation 1. In that case, the expressions would be named according to a convention and the application for Operation 2 would reference the established expression name while creating the feature.

2.3.3 Step 3: Blends and Chamfers
The method for automating the creation of blends and chamfers is also based on the interface types associated with the part. Blends and chamfers are commonly used to facilitate the assembly of mating parts and are applied to easily predicted edges, especially when the interface type is known. The

method used in this step of Operation 2 queries the edges of each part and determines which edges intersect the control structure $I_{ij}(C_i^2)$. Then based on the rules for each interface type, blends and chamfers are applied. The rules for whether a chamfer or a blend would be inserted are based on the specific vertex of the interface and could reference either the expressions created by the interface object functions or the values retrieved from a GUI. Here is an outline of the procedure.

For all parts $C_i^2 \in A^1$ $(0 \le i \le n)$

For all edges $e_b \in (C_i^2)$ $(0 \le b \le l)$

For all key vertices $v_a \in I_{ij,k}(C_i^2)$ $(0 \le a \le m)$

Calculate minimum distance d_{min} between v_a and e_b

If d_{min} < tolerance

Create Chamfer or Blend

Stop looping vertices

Fig. 6 illustrates the models after Step 3 has been completed. The yellow points denote the key vertices to which chamfers or blends have been applied.

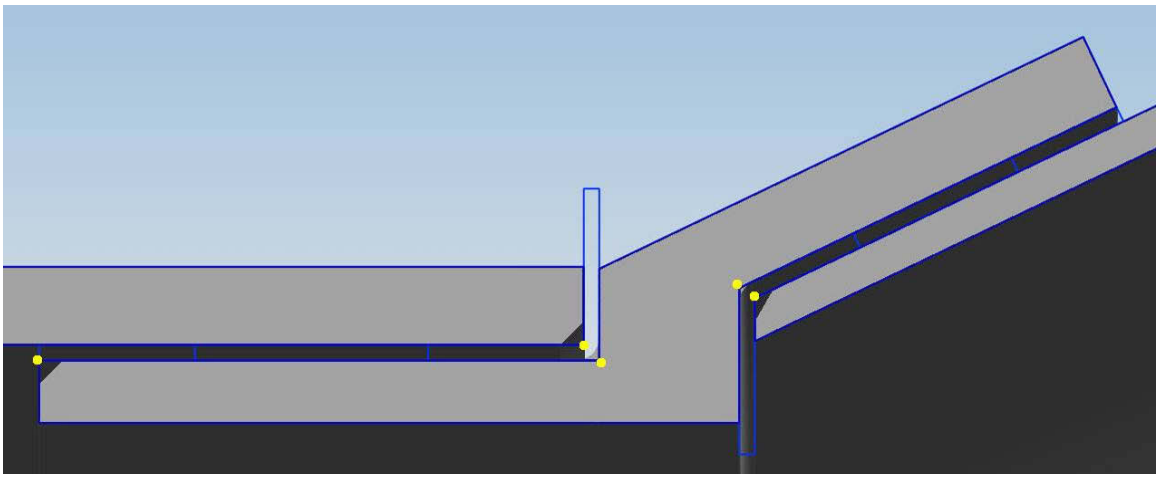

Fig. 6: Operation 2 step 3 complete.

2.3.4 Step 4: Fastener Detail Features

The method for adding fastener hole features is very similar to the blends and chamfers method, except instead of cycling through the vertices associated with the interface control geometry, it cycles through the fastener control geometry for each interface object. Again, the hole features and pattern features reference the expressions which were created by the Interface Object class member functions. The algorithm goes as follows (See Fig. 7 for the final results):

For all parts $C_i^2 \in A^1$ $(0 \le i \le n)$

For all center lines $cL_a \in I_{ij,k}(C_i^2)$ $(0 \le a \le m)$

Insert Hole Feature

Name Hole Feature according to convention

For all features $f_b \in (C_i^2)$ $(0 \le b \le l)$

If feature type and name match convention for an interface hole

Create hole pattern

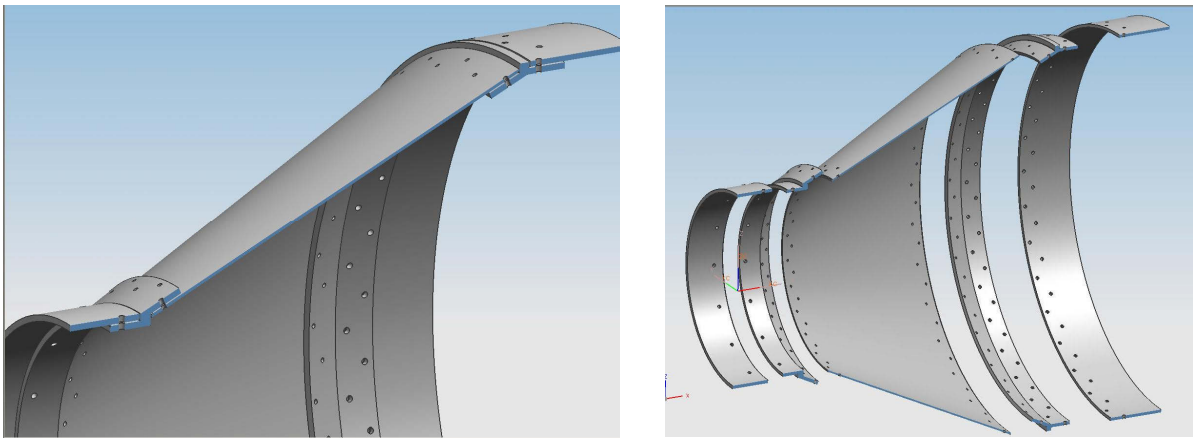

Fig. 7: (a) Completed assembly (b) exploded section view of completed assembly.

3. RESULTS

To determine whether the objectives of this research were met, and to what extent they were or were not successful, three elements of the objectives will be evaluated:

1. the range of designs supported by the framework
2. the time savings observed
3. the proportion of the design left open to the engineer

3.1 Range of Supported Designs

As a theoretical framework, the methods developed in this paper will work for any interstage assembly design. In practice, there are certain limitations. The current implementation of these methods is capable of supporting any interstage design, subject to these limitations:

Interface Manager Limitations

- The interfaces must be either cylindrical or conical bolted flanges and each type can be placed in one of four orientations
- The fastener pattern must be either a single row or a double offset row pattern
- The distances from the fastener centerlines to the flange edges and between the fastener rows are predetermined but may be changed later interactively.
- The axial position of any joint must be at least nine diameters from the origin.
- The minimum radius for any interface is three inches.
- An assembly component must contain either one or two interfaces.

Component Cross Sections Limitations

- All cross sections must contain closed loops and must not self-intersect.
- The thickness of any bolt flange must be less than five inches.
- All cross sections must extend at least the entire length of the control structure's flange.

Detail Features Limitations

- The chamfer and fillet dimensions cannot be pre-specified. They may only be changed after running the application.

Considering that this implementation of the framework provides eight combinations of interface types and orientations and allows each component to be associated with either one or two interfaces, there are 72 distinct combinations of interfaces possible for each component in the assembly. With more development, this framework would also work for other types of products that use extruded cross sections and different interface types.

3.2 Time Savings

To determine the value of the proposed methods, engineers were asked to perform a set of modeling tasks using both the traditional approach and the approach implemented in this paper. Each task correlated with one of the method's main Operations. For each engineer, the number of key-strokes and mouse clicks, and the completion time was recorded for each method. The results from each task were compiled to estimate the time and effort required to model an entire interstage assembly.

3.2.1 Task 1: Interface Manager

In task 1, the engineer must define the control structure for one cylindrical interface and one conical interface including the hole location sketches. He/she must then add a new component to the assembly, create the linked geometry features, and create a fully constrained sketch of the part cross section. Tab. 1 lists the results of the three engineers for both methods and the comparison between the two methods.

3.2.2 Task 2: Detail Features

In task 2, the engineers must make a revolve feature from the cross section, add chamfer and blend features, and create the hole extrudes and patterns. See Tab. 2 for Results from the three test subjects.

3.2.3 Entire Assembly

The results from the three engineers can be used to extrapolate an estimate for the effort required to create the parametric models for an entire interstage assembly. An example layout of a fictitious interstage assembly is depicted in Fig. 2. It includes four interfaces, and five components. Tab. 3 lists the estimated average results for this configuration based on these assumptions:

1. Creating the four interfaces will take twice as much effort as measured in Task 1.
2. Inserting the detail features on all five components will require five times the effort measured in Task 2.

Test Subject	Traditional Method			Proposed Method			Percent Difference		
	Key-strokes	Mouse Clicks	Time (min.)	Key-strokes	Mouse Clicks	Time (min.)	Key-strokes	Mouse Clicks	Time
1	864	717	45.27	84	99	5.48	90.60%	86.19%	87.89%
2	1236	947	61.78	89	95	4.67	92.80%	89.97%	92.45%
3	1226	795	65.00	54	148	6.87	95.60%	81.38%	89.43%
Average	**1108.7**	**819.7**	**57.4**	**75.7**	**114.0**	**5.67**	**93.00%**	**85.85%**	**89.92%**

Tab. 1: Task 1 completion statistics.

Test Subject	Traditional Method			Proposed Method			Percent Difference		
	Key-strokes	Mouse Clicks	Time (min.)	Key-strokes	Mouse Clicks	Time (min.)	Key-strokes	Mouse Clicks	Time
1	220	201	14.28	0	2	0.12	100%	99.00%	99.18%
2	72	178	10.48	0	3	0.10	100%	98.31%	99.05%
3	31	199	9.68	0	2	0.10	100%	98.99%	98.97%
Average	**107.6**	**192.7**	**11.48**	**0**	**2.3**	**0.11**	**100%**	**98.77%**	**99.07%**

Tab. 2: Task 2 completion statistics.

3.3 Openness of Design

The primary aspect of the application that allows for flexibility in the design is the *InterfaceManager.* It allows the engineer to define any number of component interfaces, and provides a substantial number of interface types to choose from as mentioned in section 3.1. It also lets them create an assembly with any number of components. Another key to providing openness is the fact that the designer creates the component cross section sketches. This permits virtually unlimited variations in

the design of the part bodies and is especially valuable for the actual interstage component since many different types of cross sections are used.

	Key-strokes	Mouse Clicks	Time (min.)
Traditional Method	2755.4	2602.9	172.2
Proposed Method	156.58	239.5	11.83
Percent Difference	**94.3%**	**90.8%**	**93.1%**

Tab. 3: Estimated results for entire assembly.

Decision	Score
Assembly Layout	
number of components	3
topology of each part's cross section	3
Dimensions of each part's cross section	3
Chamfers and fillets	2
Joining method for each part-to-part interface	
Interface type	1
Interface position	3
fastener size	2
fastener pattern	1
dimensions of the joint	2

Tab. 4: Openness scores for primary design decisions.

Overall the fact that the models are entirely parametric means that any of the dimensions and expressions can be changed to suit the specific needs of the designer. Therefore, even the portions of the geometric design that are hard coded into the application, such as the flange length proportions or the chamfer and fillet sizes, can be modified after running the application. To quantify the level of openness provided by the application, each of the primary decisions which must be made by the designer has been assigned a score from 1 to 3 with the following significances (See Tab. 4):

1. The designer can choose from a finite set of options
2. The designer can modify the parameters of the decision
3. The designer can change anything about the decision within normal design limits

3.4 Discussion of Results

The primary difficulty in creating parametric design tools is balancing the tradeoffs between speed and design freedom. The results presented in section 3 have shown that the methods developed in this paper are able to decrease the required time and effort by more than 90% while still leaving a majority of the primary decisions open to the designer.

Although these methods have been evaluated with the design of rocket interstage assemblies, they have potential application in any assembly dominated by similarly oriented 2½ dimensional components. A primary advantage of these high-level programmatic operations over other design automation tools, such as UDFs, is that these methods are able to operate on multiple components. Therefore, they can create the inter-part associativities and expressions that are necessary in parametric assembly modeling. In addition, they are able to generate much larger sets of geometry since UDFs cannot use their own entities as inputs to their other features e.g. A UDF would not be able to contain an offset surface feature and a feature that trims said offset surface, since the user would not be able to identify the input surface to the trim feature.

Another important advantage of the methods presented here is that they drastically reduce user error and can be executed by novice engineers or even technicians. During testing, many of the manual operations had to be repeated or corrected because the wrong input geometry was selected, or because input values were wrong. Programmatic methods do not have these problems. There were still some user errors while testing the programmatic methods, but they were usually due to unclear instructions and were much less frequent. In addition, the associative links between components assure that design changes are properly propagated throughout the assembly automatically. This further reduces design time and modeling errors.

One of the drawbacks of these programmatic methods is the time required for development and maintenance. It takes significantly more effort and expertise to write the code than it does to generate models interactively. For this reason, the proposed methods must be used in circumstances where the upfront fixed costs and intermittent maintenance costs can be justified by long-term savings and/or increased market competitiveness.

4. CONCLUSIONS

The objectives of this research were to show that high-level, product type-specific operations can accelerate the design of a wide range of rocket interstage components and assemblies and that these operations will decrease the design time without impeding innovation.

In section 2, a method was developed to define the assembly layout using a framework of C++ classes and user interfaces called the *InterfaceManager*. While this theoretical framework was capable of supporting any assembly layout, the framework that was actually implemented was limited to eight types of interfaces. Section 3 demonstrated that the *InterfaceManager* still supported a very large number of interface combinations even with these limitations and was able to create the interfaces around 90% faster than by using the traditional method. From these results we can conclude that product type-specific operations can greatly reduce modeling time of assembly layouts and can be flexible enough to support wide spectrums of designs.

Section 2 also discussed methods for creating the detailed features on each part in the assembly including the revolve features, chamfers, fillets, and hole patterns. These methods resulted in more than a 98% reduction of modeling time and effort. Since the inputs for these detail features were defined by the *InterfaceManager*, no effort was required of the user to detail the parts. These results prove that CAD design can be streamlined extensively using high-level operations.

5. REFERENCES

[1] Bidarra, R.; Idri, A.; Noort, A.; Bronsvoort, W. F.: Declarative user-defined feature classes, Proceedings of the 1998 ASME Design Engineering Technical Conferences, CD-ROM, September, 1998 13–16. Atlanta, GA, USA, New York: ASME.

[2] Delap, D.; Hogge, J.; Jensen, G.: CAD-centric creation and optimization of a gas turbine flowpath module with multiple parameterizations, Computer-Aided Design & Applications. 3(1-4), 2006, 175-184.

[3] Emch, F.: Impact of System-Level Engineering Approaches on the Airframe Development Cycle Via Integration of KBE with CAD Modeling and PDM, RTO AVI Symposium. April, 2002.

[4] Hoffman, C. M; Joan-Arinyo, R.: On User-defined Features, Computer Aided Design, 30(5) 1998, 321-352.

[5] Jankowski, G.: Solid Thinking: Using Functional Features to Build Plastic Parts, Cadalyst Nov. 15, 2005. Retrieved on 12/10/07 from: http://manufacturing.cadalyst.com/manufacturing/article/arcticleDetail.jsp?id=197017.

[6] Lamarche, B.; Rivest, L.: Dynamic Product Modeling with Inter-Features Associations: Comparing Customization and Automation, Computer-Aided Design & Applications, 4(6), 2007,877-886.

[7] Ledermann, C.: Associative parametric CAE methods in the aircraft pre-deisgn, Aerospace Science and Technology, 9(7), 2005, 641-651.

[8] Mosca, F.; Di Martino, C.; Aleixos, N.: Complex CAD project management by the means of designing criteria control tools, Deployment of a vehicle gearbox archetype with the aid of WAVE by UNIGRAPHICS, XII ADM International Conference, September 2001.

[9] Shah, J. J.; Ali, A.; Rogers, M.T.: Investigation of declarative feature modeling, Proceedings of the ASME 1994 Computers in Engineering Conference, ASME, NewYork, 1, 1994, 1-11.

[10] Scott, N. W.: High-Level Product Type-Specific Programmatic Operations for Streamlining Associative Computer-Aided Design, M. S. Thesis, Brigham Young University, 2008

[11] Tang, M.; Wen, Y.; Mi, X.; Dong, J.: Parametric modeling with user-defined features, Computer Supported Cooperative Work in Design, The Sixth International Conference on, 12-14 July, 2001, 207 – 211.

[12] Venkataraman, S.; Shaw, J. J.; Summers, J.: An investigation of integrating design by features and feature recognition, International Conference FEATS, 2001.