49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition
4 - 7 January 2011, Orlando, Florida

AIAA 2011-159

A More Efficient Conceptual Design Process Using the RAGE Geometry Modeler

Dejapong L. Suwaratana[*] and David L. Rodriguez[†]
Desktop Aeronautics, Inc., Palo Alto, CA, 94303

A case is made that introducing CAD-based geometry prematurely in the aircraft design process actually hinders multidisciplinary design. Consequently, a design process which allows a designer to quickly build and analyze aerospace configurations without CAD or computational meshing specialists is proposed. This allows higher fidelity analyses to be applied at conceptual design and decreases the turnaround time for trade studies and optimizations. The basic premise of conceptual geometry modeling is presented and discussed. Key features and capabilities which are essential for a successful geometry modeler are identified. Desktop Aeronautics' conceptual geometry modeler, RAGE, is described, with emphasis on recent progress and advances in the tool. A preliminary graphical-user-interface is presented along with an application programming interface. Extensibility of the RAGE modeler is discussed along with the importance of this feature. The applicability of RAGE to simple trade studies is demonstrated in an example problem. The precision of the RAGE modeler is also demonstrated on a complex aircraft design. Future plans for the geometry modeler are also discussed.

I. Background

In the conceptual phase of aircraft design, there is often a wide range of geometric configurations that the designer wishes to consider, analyze, and assess. A designer’s intuition and creativity are crucial to a successful conceptual design, yet are easily inhibited by the modeling and analysis process itself, particularly if high-fidelity analysis is warranted. If the time between inception and assessment of a candidate design is relatively long, a designer may be forced to limit the number of concepts considered. New data revealed in a design iteration may suggest or require significant changes to the geometry configuration. Since the majority of considered design configurations will eventually be discarded, an ideal conceptual design process should involve a highly efficient and easy to use geometry modeler. Significant changes should be allowed without tedious re-work of the geometry model. Exporting the model for analysis should require little effort and have the capability to be automated. Therefore, any modeling application used in this process should be flexible and not restrict the designer’s choice of user interface, analysis method, and design optimization environment.

Most current design processes in the aerospace industry rely heavily on computer-aided-design (CAD) applications. As a general design tool, CAD is a powerful geometry modeler capable of precisely representing practically any shape. Unfortunately, this generality is realized at the cost of requiring specialized training to efficiently and effectively use the software. Also, a CAD model is often not easily modified, especially if significant re-work of the geometry is necessary. This can lead to the premature freeze of an aircraft design due to employing excessively high-fidelity geometry modeling too early in the design process. Parametric CAD modeling could address this issue, but creating a well-designed parametric CAD model is difficult. Not only does it require an extremely skilled operator, but also the designer must have some insight into what parameterization may ultimately be desired. Modifying a parametric CAD model is often more difficult than modifying a conventional CAD model. Significant changes, such as adding or removing cross-sections, may invalidate the parameterization in some cases.

In most organizations, CAD modeling (whether it be parametric or conventional) is often assigned to a dedicated CAD specialist. Since the conceptual designer is usually not a CAD specialist, the latter must communicate his design intent to the former, often using sketches, dimensions, and word of mouth. When the CAD specialist interprets the sketches, any part or detail not precisely defined usually requires judgment and artistic license. This interpretation may result in loss of design intent and indeterminate iteration between the CAD specialist and designer to obtain a satisfactory result. Because this process is iterative, CAD operators often must recreate similar geometry

[*] Design Engineer, AIAA member.

[†] Senior Engineer, Senior AIAA member.

Copyright © 2011 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

over and over again, which is highly inefficient. Even after the desired geometry is obtained, creating a computational analysis mesh often requires yet another trained specialist. The desired meshes for different analysis methods may require further modification of the geometry model and therefore additional communication between the meshing and CAD specialists. Consequently, analysis results may not be available to the designer for days or even weeks after a design was conceived. Changing the design would require repeating the entire process. Contrary to the ideal claimed, the CAD-based design process has a high geometry-creation cost that often increases as the design progresses. Rework occurs often, and the effort must often be delegated to several specifically trained individuals.

In some organizations, the process above is only put through one iteration. During the conceptual design phase, an aircraft is sized using rudimentary handbook methods usually based on historical data. For instance, this process might consist of a designer using spreadsheets and other very low-fidelity analyses. Once the designer is satisfied, the geometry characteristics are relayed to a CAD specialist who builds the geometry model. At this point, the geometry may be practically frozen and sent to the separate disciplines for analysis, slight tweaking, and detailed design. This of course eliminates the chance of any true multidisciplinary design or optimization after the CAD model is built. This process may be successful for building the same type of airplane over and over again, but is far from ideal. A truly optimal design is rarely discovered. Of course, if the design is revolutionary where no historical data is available, this process will most likely lead to a poor design because the handbook methods result in incorrect assessments.

Many high-fidelity analysis methods available today are quite fast and easy to use. For instance, it no longer takes weeks to obtain a good Euler solution on a complete aircraft configuration. Also, many modern analysis methods no longer require specialized training to obtain accurate and meaningful results. It is now becoming reasonable to employ high-fidelity methods in conceptual design. This in turn allows for multidisciplinary optimization methods to be applied with these high-fidelity analysis methods. With the advancement of all these methods, the conceptual designer has quite a powerful set of tools available. What was once merely an academic exercise on notional aircraft can now actually be used by industry in the conception of new aircraft. In fact, high-fidelity-based conceptual design methods are essentially necessary when designing revolutionary aircraft such as supersonic jets with extensive laminar flow and hybrid-wing-bodies.

But the one tool that is also vital to this process that has not advanced sufficiently in terms of efficiency and ease of use is the CAD modeler. CAD software is built to model anything, from aircraft to automobiles, from bridges to buildings, and from air conditioners to heat pumps. CAD allows the designer to specify precise manufacturing lines and tolerances for each and every part of a system, including nuts and bolts. But this incredible capability comes at the cost of requiring specialized training to be able to fully exploit the tool. As capable as CAD software has become, modeling an aircraft in CAD during the conceptual design phase is simply overkill.

Consider a typical conceptual aircraft design process. At first, the designer only knows the basic properties such as the wing span and aspect ratio. After some basic sizing of the aircraft, the designer will then choose airfoils, a wing sweep, and a twist distribution. At this point, the aircraft concept has an actual outer mold line which could be analyzed and assessed even with high-fidelity tools. In fact, in order to successfully identify an optimal design, the designer would need to analyze and assess many, many outer-mold-line shapes. This is especially true if optimization schemes are employed. But using CAD to generate these many outer-mold-lines is extremely cumbersome. Some engineers do currently use parametric CAD to generate geometry in optimization scheme, but they often find the time necessary to generate the geometry takes longer than the high-fidelity analysis methods. CAD is simply not ideal for conceptual aircraft design.

II. Methodology

A different process is proposed to resolve the aforementioned deficiencies in traditional CAD-based conceptual design that allows geometry to be quickly created, meshed, and analyzed. Instead of requiring the designer to communicate design intent among separate specialists as diagrammed in Fig. 1, he is given a set of tools that allows him to rapidly perform each task himself. A parametric geometry modeler utilizing intuitive aircraft design parameters replaces CAD in generating aircraft geometry. An entire configuration is generated by building aircraft components that are based on intuitive design parameters such as wing span, fuselage diameter, and airfoil thickness. The designer can visualize the geometry

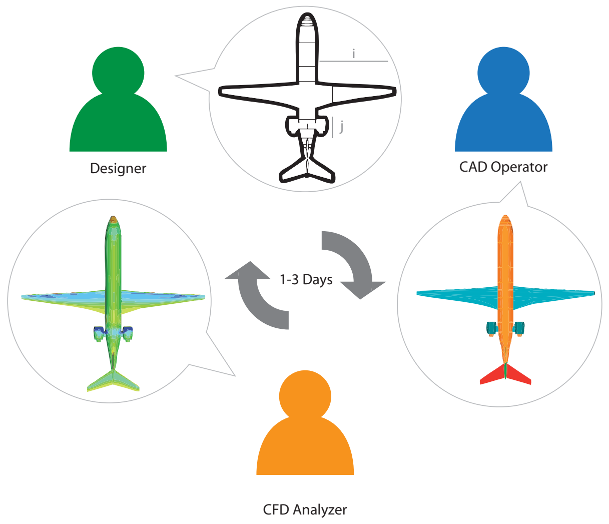


Figure 1. Typical CAD-based conceptual design process.

resulting from the defined parameters in real time and therefore no loss of design intent occurs due to miscommunication with other engineers. Once the model has been generated, preparing an analysis mesh is automated. After analyzing the design, the designer can quickly modify the geometry and reanalyze a new design. Fig. 2 portrays an iteration of this proposed design process.

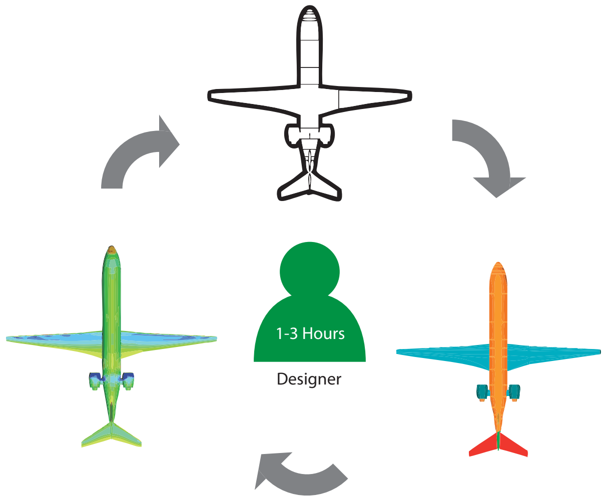


Figure 2. Conceptual-geometry-based design process.

Another advantage of this new process over the traditional CAD-based method is the training time. Because the conceptual geometry tool uses intuitive parameterizations of aircraft geometry, the learning curve for the tool is relatively low, especially when compared to the very steep learning curve of CAD software. This means the designer not only becomes efficient with the tool quickly, but also is able to use the tool again even after months of not using it at all.

But perhaps the biggest advantage of using a conceptual geometry tool in lieu of CAD is the design cycle time. What typically takes days with a designer and two specialists (Fig. 1) can be accomplished in a matter of hours by the designer alone (Fig. 2). Reducing the design cycle time can either decrease the total amount of time devoted to conceptual design, or allow the designer to analyze many more concepts to perhaps improve the final design. It also allows for more successful numerical optimizations to be implemented early in the design process, which could both improve efficiency and the performance of the final design.

In order for this proposed process to be successful, the conceptual geometry modeler must incorporate certain capabilities and features such as a simple user-interface, rapid geometry generation, intuitive parameterization, and automated meshing for various analyses. A handful of conceptual geometry modelers exist that incorporate many of these features including VSP[1], PAGE[2], and GGG[3,4]. These modelers are all being used in various settings, but mostly in research and academia. Conceptual geometry modeling has not yet been adopted extensively in the industry, with the exception, perhaps, of Boeing who developed GGG. Otherwise, most other large aerospace companies continue to use CAD for all phases of design.

Ref. 5 describes the RAGE (Rapid Aerospace Geometry Engine) conceptual geometry tool which attempts to incorporate all of these basic, necessary capabilities. As originally detailed in Ref. 5, complete aircraft geometries are designated geometry *configurations* when using RAGE. These are in turn collections of parametrically defined geometry *components*. Geometry components are created, in general, by stacking and mathematically lofting cross-sections (or *subcomponents*) along a defined axis. For example, specifying airfoil types at various stations along a spanwise axis creates a wing. Likewise, defining cross-section shapes at locations along a longitudinal axis creates a fuselage. This basic process is depicted in Fig. 3. All subcomponents, such as airfoils and cross-sections in these examples, are mathematically defined by user-provided parameters. For example, an airfoil set of parameters may include airfoil family, thickness, maximum thickness location, and camber shape. Note the definition of the airfoil is precise, not an approximation. Also, the number of parameters required to describe the airfoil is usually quite small compared the number of control points that would be required for a precise CAD B-spline representation. Similarly, an elliptical fuselage cross-section is defined by position, width, and height parameters. Loftings between cross-sections can be controlled in fine detail. Each component defines a watertight surface geometry which can later be discretized and intersected for analysis.

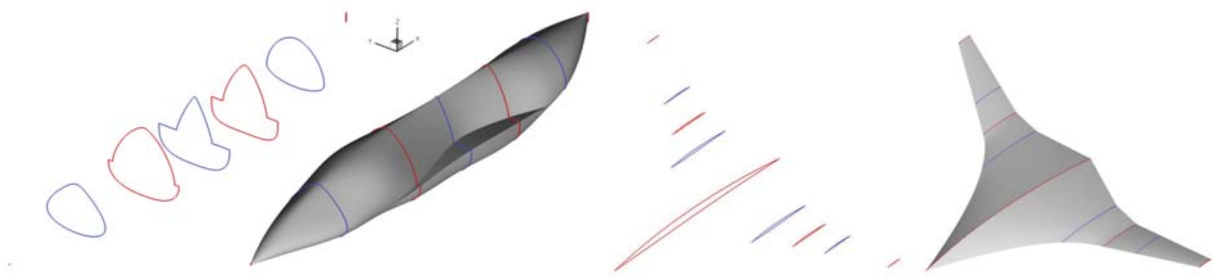


Figure 3. Examples of RAGE fuselages and wings built by lofting sections.

RAGE provides several types of parametrized components, and is therefore capable of generating a very diverse set of geometries, including most aircraft configurations. Fig. 4 shows a few examples of RAGE-generated geometry models. If an externally-generated part must be included in a model, a discretization of that part may be included as a component of a configuration. This component can subsequently be modified and included in a full configuration analysis mesh. This capability is useful in cases where a new component is to be designed for an already existing geometry, or geometry is to be designed in relation to an existing aircraft.

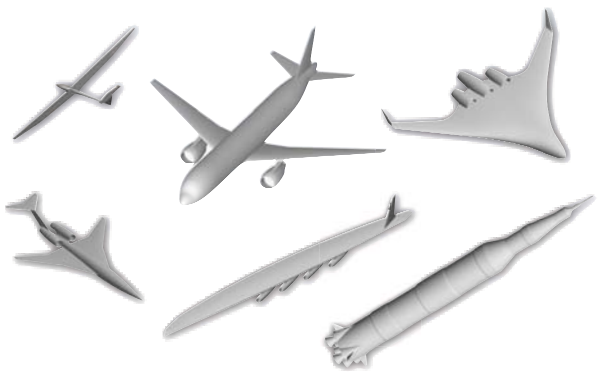


Figure 4. Sample RAGE-generated geometries.

Since Ref. 5 was published, several improvements and additional features have been added to RAGE. These features are considered essential to a quality geometry modeler, especially if widespread adoption in the industry is to occur. The following sections describe these features and reasons why the authors consider them so important.

A. User Interface Requirements

A good user interface is absolutely critical for widespread adoption of a software product. The interface must be flexible enough to be useful in any framework and by any user. For a conceptual modeler, the beginner should be able to quickly build basic aircraft geometries without extensive training. This is especially important for conceptual aircraft designers who might use the geometry modeler sporadically with large periods of time between uses. Being able to quickly understand and use the interface is key for this usage pattern. For this type of user a simple, self-explanatory, and intuitive graphical-user-interface would be appropriate.

On the other hand, the so-called "power user" should be able to use all the features available without being inhibited by a user-interface that might require a great deal of interaction. For this type of user, the interface might be spartan but more powerful. For instance, all the aircraft parameters would be stored in something that resembles a spreadsheet rather than a menu-and-slider graphical interface. Being able to edit that text file directly and still see the effects in real time would be very useful for this type of user.

A third type of interface is necessary to fully exploit the capabilities of a conceptual geometry modeler. Integrating a geometry modeler with other design tools or even a design optimization framework requires a completely different kind of interface. For this application, the modeler must be accessible programmatically. Therefore a good geometry modeler should provide an application program interface. This is discussed further in a section that follows.

The RAGE modeler provides some form of all three of these types of interfaces. The most utilized and mature interface is the text-file based interface. This would be appealing to the power-user. A human-readable text file is edited and the modeler instantly reacts to any changes in the file, displaying the geometry in a separate window. A screen shot of this interface is given in Fig. 5. Like most applications that display geometry, the model can be rotated, magnified, etc. The geometry can also be shown as a solid surface, a wireframe, or even a cloud of points. The purpose of this simple interface is to present the power user with as much information as possible and to provide a quick and direct way to change the parameters that define the geometry. Collapsing windows or parameters controlled by sliders may present a better organization of the parameterization, but it means the power user will have to perform many mouse clicks just to get to a single parameter. If all the data is in a text file, the power

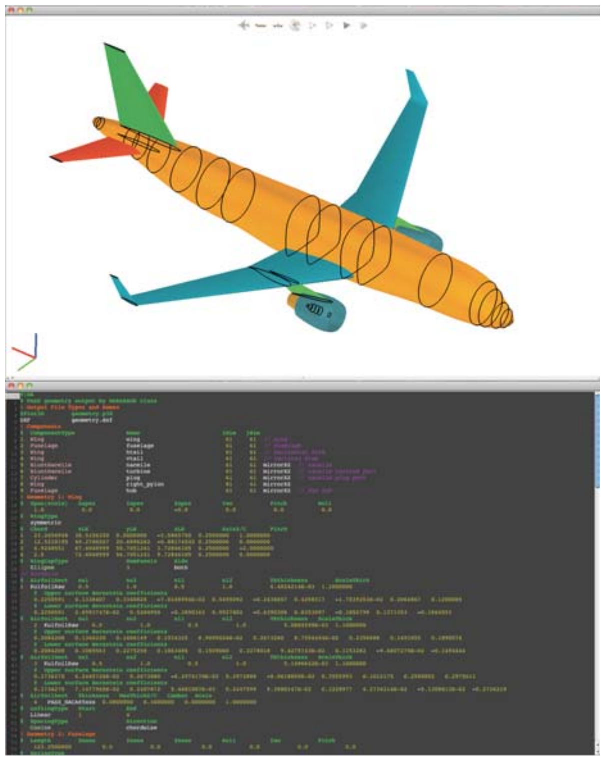
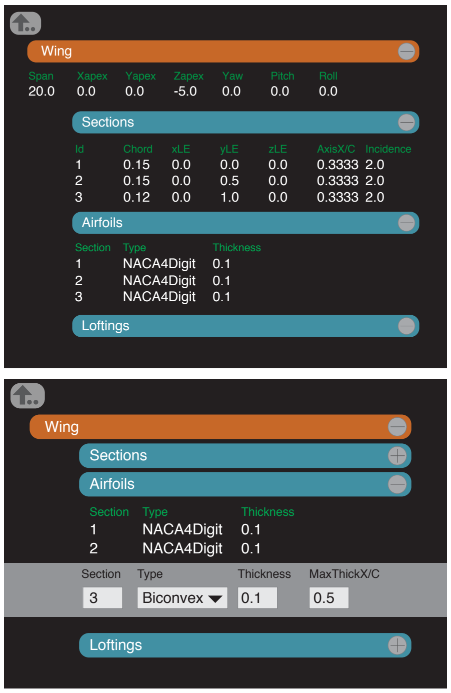


Figure 5. Text-file-based user interface for RAGE.

user can just change the parameter values directly to exactly what is desired.

Of course, the beginning or sporadic user would easily be overwhelmed by the table of raw data shown in Fig. 5. This user would rather have a more assistive interface. Under development for RAGE is a graphical input method that does not display the actual input file, and instead displays components in a collapsing tree format. Components and subcomponents are represented as branches in the tree graph. Parameters form the leaves in each branch and can be directly edited. Navigating through the tree will highlight the corresponding component or section in the rendering. Clicking on a leaf node creates input fields where parameters can be edited. Branches can also be isolated, hiding irrelevant information in the interface on a subset of the configuration. While isolating a component, the visualizer window will fit the component to the rendering screen, and parent and sibling branches will be hidden in the graph view. For example, Fig. 6 shows a branch of the configuration tree representing a wing. The grey arrow in the left top corner indicates to the user that the editor is currently in isolation mode for the wing component, and that hidden branches exist above or on the same level. Sub-branches of this wing include sections, airfoils, and loftings. Clicking on the negative symbol to the right of the branch title hides the parameters for that element. By selecting a row of parameters, a grey highlight bar appears, allowing the user to edit the values for that parameter as shown on the bottom of Fig. 6. Once changes have been confirmed, the resulting geometry will be rendered in the transformable view as in Fig. 5. This tree-based interface will prove particularly useful for users new to RAGE or when a geometry is to be created from scratch where no template is available.


Wing

Span	Xapex	Yapex	Zapex	Yaw	Pitch	Roll
20.0	0.0	0.0	-5.0	0.0	0.0	0.0

Sections

Id	Chord	xLE	yLE	zLE	AxisX/C	Incidence
1	0.15	0.0	0.0	0.0	0.3333	2.0
2	0.15	0.0	0.5	0.0	0.3333	2.0
3	0.12	0.0	1.0	0.0	0.3333	2.0

Airfoils

Section	Type	Thickness
1	NACA4Digit	0.1
2	NACA4Digit	0.1
3	NACA4Digit	0.1

Loftings

Wing

Sections

Airfoils

Section	Type	Thickness	MaxThickX/C
1	NACA4Digit	0.1	
2	NACA4Digit	0.1	
3	Biconvex	0.1	0.5

Loftings

Figure 6. Tree-based graphical user interface for RAGE.

Again, this interface is not designed to replace the text file interface but rather to compliment it. One can imagine how inefficient this interface would be for a power user considering all the clicking of the mouse that is necessary to adjust just one parameter. A modeler that only provides this kind of interface could very quickly become unappealing to the advanced user.

B. Extensibility

A general modeling tool such as CAD allows the creation of practically any geometry. This is a distinct advantage that CAD has over a conceptual geometry modeler, and one that designers in the industry will always consider before adopting the modeler. On the other hand, this capability often results in a complex interface and an overwhelming array of options. This means the geometry modeler must be much simpler than CAD to prove more useful and appealing. This is an important tradeoff which is not always clearly determined. One thing is certain: it is impossible to create a geometry modeler that will provide parameterized geometry components for every possible aerospace configuration. If such an application were to exist, it would become at least as complicated as a CAD package and therefore self-defeating. Then again, if the modeler were too limited, many potential users would find it inadequate for their unique geometries and promptly abandon the modeler. The only way to maintain simplicity and yet not restrain its capabilities is to make the modeler extensible. The user has to be able to build and add his own parameterized geometry models easily. Geometry modelers that require users to request the developers for additional features will quickly become less useful.

Consequently, RAGE was designed from the ground up to provide extensibility. Object-oriented programming makes this possible and rather straightforward. If a new component or subcomponent that cannot be modeled is needed, a designer can create a single Java class that either extends an existing component or defines a brand new one. The new class defines the parameterization and then provides the necessary algorithms to build the geometry based on this parameterization. Because of the way RAGE is built, user-developed RAGE components are never interface-specific and display consistently in all user interfaces. To add the new component to RAGE, the class file is simply added to the appropriate package folder after which it can then be used within any interface without the user having to program any interface elements.

This capability is a very powerful feature. While RAGE is designed to be able to model most aerospace geometries, sometimes additional detail is necessary for unconventional geometries. For instance, a new component had to be developed to properly model and parametrize the strake and wing of the Aerion business jet[6]. Perhaps a designer has developed a new airfoil and wishes to use it on a wing. The designer simply has to create a short new airfoil class (on average, about 5-10 new lines of actual code) that provides the methods for computing the upper and lower surfaces at a given chordwise location. The class is added to the airfoils package and is instantly available to all interfaces. The same holds true for cambers, wing caps, fuselage cross-sections, lofting algorithms, fan face hubs, and virtually all other RAGE parts. Without the ability to extend what is already available, designers would quickly become frustrated whenever a desired geometry seriously challenges the inherent capability of RAGE. Furthermore, as the RAGE-user base grows, any user-developed components or subcomponents could be added to a global repository for all users to share.

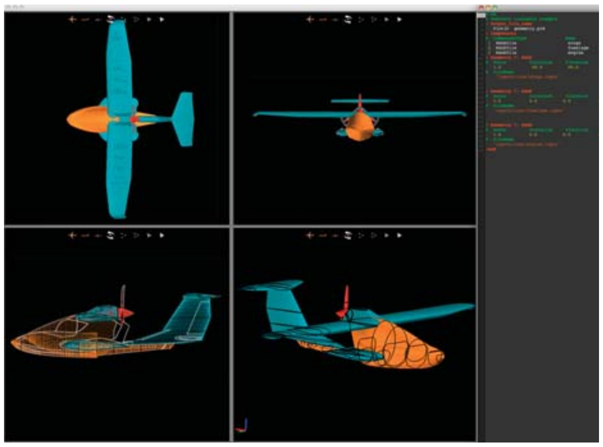


Figure 7. Extended version of the original RAGE interface developed to display a provided 3-view as an underlying canvas.

C. Application Programming Interface

As mentioned above, the most powerful way to employ a conceptual geometry modeler is in a trade study, optimization tool, or any other design framework. Ideally a geometry modeler would be able to interface with all of the many diverse applications in the industry. The best way to accomplish this is to provide an Application Programming Interface (API). This would allow users to couple the modeler to both modern and legacy codes and fully exploit their new tool.

Consequently, an advanced RAGE API is currently under development. This API allows the designer to easily integrate RAGE into any analysis or design framework. For example, through the API, RAGE was integrated into PASS[7] to design a commercial airliner that exploits natural laminar flow[8]. Users also have the capability to extend the API to function with their own extended component and subcomponent classes.

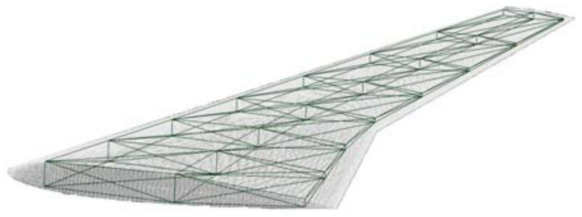
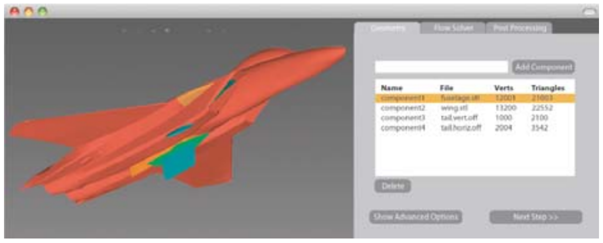


Figure 8. Finite element analysis model generated using the RAGE API. The graphical interface for the application was also developed using RAGE libraries.

This API is not limited to geometry components. For example, for the light sport aircraft design case in Section III, an extension of the visualizer was written to allow an existing three-view of the aircraft to be the rendering canvas as shown in Fig. 7. As the geometry was created, the 3-view overlay assisted in choosing parameters so that the geometry model matched the existing aircraft.



Name	File	Verts	Triangles
component1	fuselage.stl	12001	21003
component2	wing.stl	13200	22552
component3	tail.vert.off	1000	2100
component4	tail.horiz.off	2004	3542

Figure 9. Graphical interface for Cart3D developed using RAGE interface libraries.

The API also allows external tools to be used within RAGE. Currently, a structural analysis tool is being developed alongside RAGE that creates structural elements within a RAGE wing component and performs a finite element analysis. The deflections are returned to RAGE, which can then reshape the wing parameters accordingly in an aero-structural analysis framework. An automated tool can then optimize both the outer mold line, and the structural model. Fig. 8 shows a mesh generated within a wing for finite element analysis. The mesh was generated through the RAGE API. The graphical interface for the analysis code also used the RAGE graphical interface libraries.

Another example of a tool integrating RAGE libraries is a new graphical interface (Fig. 9) currently being developed for the Cartesian Euler code, Cart3D[9]. Current interaction with Cart3D is through the command line using various text input files. The interface in development will allow users to graphically assemble and transform geometry discretizations and post-process the result. The pre-processing portion of the tool uses RAGE file readers, file writers, and meshers to prepare the geometry for analysis. Components can be scaled, translated, and rotated. Diagnostics are then performed on the final discretization using RAGE libraries to make certain the surface mesh is acceptable to the Cart3D volume mesher. The RAGE graphical interface can then be used to view and manipulate the resulting configurations, as well as visualize contours of flow data from the Cart3D solution.

III. Applications

This section presents example applications of the latest version of the RAGE modeler. The first example shows the efficiency of performing trade studies using RAGE and the Cart3D Euler analysis tool. The second example demonstrates how RAGE can be used not only for analysis of detailed models but also for generating quality, artistic renderings. Both examples also discuss the computational performance of RAGE.

A. Trade Study Example

An initial generic glider design was created using RAGE and analyzed with Cart3D. The design was then modified twice, and the wall clock time expended was recorded. The initial geometry was created using four components, namely one fuselage component and three wing components. Thirteen fuselage and wing cross-sections were used to generate the entire geometry. Creation time using the input file and visualizer interface was under ten minutes. After creation of the model, a mesh was exported, components were intersected, and the configuration was analyzed using Cart3D. This geometry is shown in Fig. 10. For this first concept, the empennage is a typical low tail. The wing is straight with only a small trailing edge extension inboard and no dihedral.

This initial geometry was then modified by adding a wing section, and changing the wing lofting from a linear lofting to a smoother, splined lofting. The empennage was changed to a T-tail design and the aft fuselage was thinned. The result of these changes is shown in Fig. 11 and required less than two minutes to complete. The analysis process was repeated for this configuration.

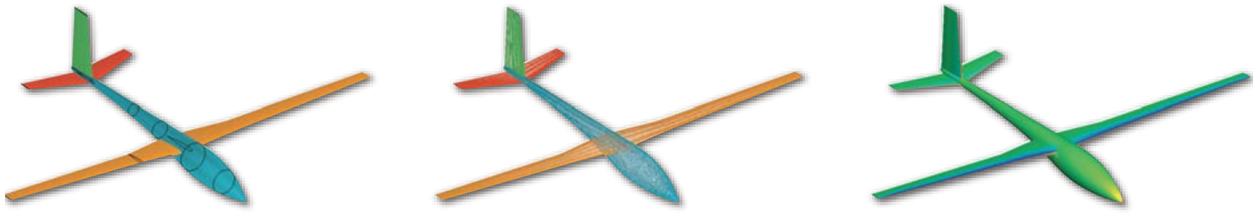


Figure 10. Initial geometry, mesh, and analysis results of glider trade study.

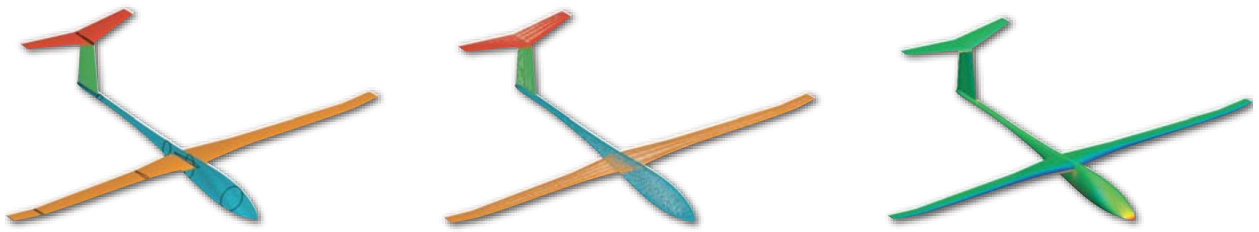


Figure 11. Second design iteration geometry, mesh, and analysis results of glider trade study.

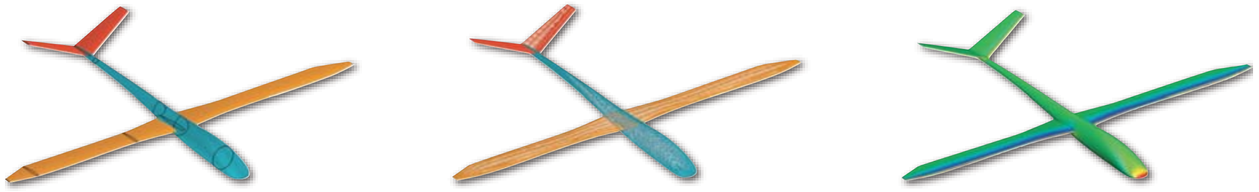


Figure 12. Third design iteration geometry, mesh, and analysis results of glider trade study.

The initial geometry was modified again in less than 2 minutes and is shown in Fig. 12. A wing section was added but this time linear lofting between sections was retained. The empennage was changed to a V-tail configuration. The fuselage nose was flattened and blunted somewhat. Again, this new geometry was analyzed using Cart3D. The total time cost for the designer was under half an hour for all three configurations. With the flow solutions running concurrently on separate remote servers, creation and analysis time for all three configurations was under an hour. This trade study example demonstrates that with RAGE the designer can run multiple configurations in a day, quickly eliminate design candidates, and not require trained CAD or meshing specialists.

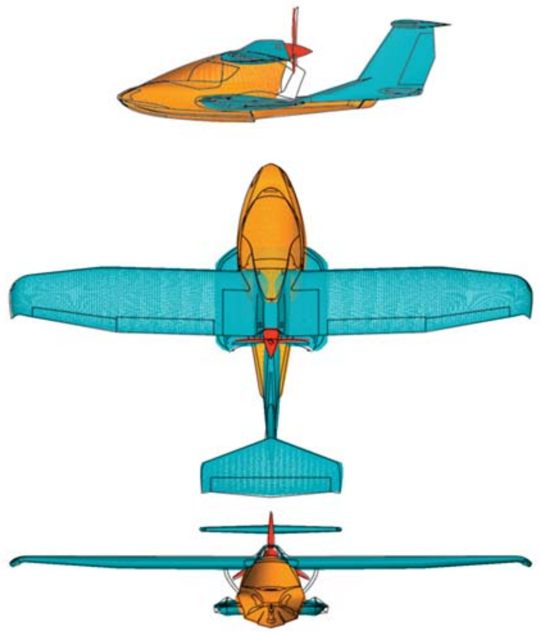


Figure 13. RAGE model of a light sport aircraft generated by matching lines from a 3-view.

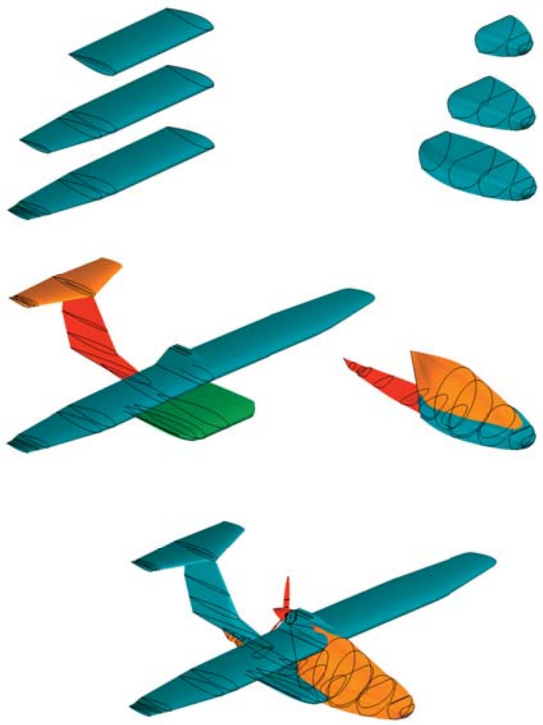


Figure 14. Progression of images portraying the modeling of a light sport aircraft in RAGE.

B. Precision Modeling

The ICON[10] light sport aircraft was modeled with RAGE for aerodynamic analysis. The purpose of this example was to demonstrate the precision attainable with parametric modeling. Starting from a three-view of the aircraft, RAGE components were created to fit within the existing lines, as show in Fig. 13. Fig. 14 portrays the progression of lofting the hull and wing. The model was constructed using a total of eight wing and fuselage components. A RAGE fuselage component was used for the hull and a symmetric wing component for the main wing. The canopy and tail boom are fuselage components with appropriate cross-sections. Wing components were also used for the floats, horizontal stabilizer, and vertical tail. Propeller blades were added using a repeated wing component. Splines and wing loftings were chosen to closely match existing outer mold lines. The final result was then intersected, meshed, and analyzed using Cart3D. Pressure contours from this solution are shown in Fig. 15.

Preparation of the geometry for analysis was automated and took several minutes. Analysis of the geometry required nearly an hour on an modern laptop with a 2.66GHz dual-core processor and 4GB of memory. Modification of the geometry from this point was not demonstrated, but could easily be accomplished. A new wing design could be added or fuselage cross-sections removed and changed within minutes. The accuracy of the model when overlaid

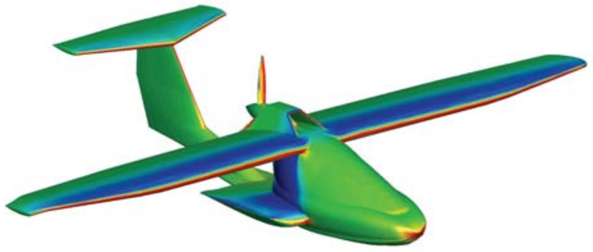


Figure 15. Pressure contours computed with Cart3D on the RAGE-generated light sport aircraft model.

Figure 16. Artistic rendering based on RAGE model of a light sport aircraft.

on the three-view demonstrates that despite the simplicity of parametric design and the reduced generality when compared to CAD, precise control of the final geometry shape is still possible.

Textures were then added to the geometry using the Blender®[11] open-source geometry tool. The model was then rendered using the GPU-based Octane Renderer[12] software. The final rendering is shown in Fig. 16. By running the renderer on a GPU, the model and lighting could be adjusted in real time and the resulting rendering only required five minutes of computation time. This combination of a parametric modeler and renderer can give a designer an added capability: the ability to generate conceptual aircraft artwork for presentations, marketing, and visualization.

IV. Future Work

Several new RAGE features described in this paper are still under development. Obviously the first task at hand is to complete development of those features and release the software. This includes the graphical interface and the API in particular. Also some features discussed in Ref. 5 have not been completed, such as the ability to output meshes for vortex-lattice aerodynamic analyses and more robust wake stitching for linear surface panel methods. These features also must be completed to allow RAGE to be useful to the majority of design engineers. But there are other features that must be included to make RAGE a complete geometry modeling tool.

While this paper professes some of the pitfalls of using CAD in conceptual design, there is no doubt that CAD is a useful and powerful tool. Once a design concept enters the preliminary and detailed design phases, geometry modelers like RAGE simply do not provide sufficient detail and flexibility. Engineers from various disciplines will require CAD loft lines to perform their tasks. For example, systems engineers need CAD models to determine how to package all the aircraft systems (avionics, cabin furnishings, etc) within the aircraft outer mold line. Of course, a final detailed CAD model will always have to be generated for manufacturing. Considering these facts, the authors do not suggest replacing CAD entirely but rather compliment the process with conceptual modelers early in the design to improve efficiency.

Since CAD cannot be replaced, then the question arises on how to transition from conceptual geometry modeling to CAD modeling. When that transition occurs is up to the designer, but the actual process is not clear. Currently the only way to transition from RAGE to CAD is rebuild the RAGE geometry completely from scratch in CAD or to translate the meshed surfaces to NURBS surfaces with other software, such as NASA's GridTool[13]. If this were a one-time procedure, it would be acceptable. The CAD model has to be generated anyway whether RAGE is used first or not. And if RAGE were used first, the CAD model would be closer to the final design anyway, perhaps eliminating many iterations of the design in a cumbersome CAD application. But ideally, the designer would prefer to continue to work on the conceptual design while other disciplines analyze his best concept produced to date. Therefore, having a smoother and more automatic transition from a RAGE geometry to a CAD model would be immensely useful.

To improve the transition from RAGE to CAD, the OpenCascade[14] libraries and geometry kernel will be used to develop a way to output CAD models directly from RAGE. Providing the conceptual designer with the ability to automatically generate a valid CAD model from RAGE would improve communication between engineers tremendously. This capability would be available to RAGE users with essentially no CAD experience, allowing for easy transfer of geometry. These tasks will constitute the primary focus of RAGE development over the next few years.

As mentioned above, another standard industry practice that must be addressed is systems packaging. In many organizations, after an aircraft is initially sized and sketched, one of the first tasks for a CAD specialist is to make certain all necessary systems (such as avionics, cabin furnishings, etc) fit inside the aircraft. This is obviously a key constraint to the aircraft outer mold line, even in the conceptual design phase. RAGE should provide a way to address this constraint if it is to be effective throughout the conceptual design process. Currently, these geometric constraints are applied by hand where conflicts are anticipated and must be updated during shape optimization if the geometry changes are large.

A better way that RAGE could address the system packaging problem is to provide a set of systems component geometries. These might prebuilt geometry models such as an avionics box or a passenger seat. Note that these models probably would not need to be parameterized. These systems components could then be used by the conceptual designer very early in the design process as constraints on the outer mold line. This would be yet another feature that would make RAGE an attractive substitute for CAD in conceptual design.

V. Conclusions

Conceptual geometry modelers gradually becoming more prolific in the aerospace industry. Geometry modelers can provide a very appealing and effective alternative to CAD, particularly for conceptual design. A single engineer can use a geometry modeler in conjunction with high-fidelity analyses to create and assess a design concept within hours. CAD-based design can take days to accomplish the same feat; this means fewer concepts are considered and the optimum design is probably not identified. Additionally, utilizing CAD early in the design cycle creates a sort of geometry inertia where the geometry definition becomes too detailed early on, and only small changes are subsequently made. Replacing the CAD system with a conceptual modeler not only increases the efficiency of the design process, but also the effectiveness of the conceptual designer and facilitates multidisciplinary trades and optimization resulting in better aircraft designs.

Since its introduction in Ref. 5, there have been several advances in the RAGE modeler features, interface, and capabilities. A preliminary graphical-user-interface has been developed which does not replace but actually compliments the old text file input method. A tree-based interface is also under development which should further decrease the training required to become a prolific user of RAGE. The extensibility of RAGE has also been improved, allowing users to easily create custom geometry components and subcomponents. This extensibility has also carried through in the graphical user interfaces allowing users to add components without having to code any part of the interface. An application programming interface is also under development which will allow users integrate RAGE with their existing design tools more quickly and easily.

The demonstrated applications of RAGE further support the claim that conceptual geometry modeling with RAGE is efficient and easy to use. RAGE parametric models have also been shown to be able model complex aircraft precisely, certainly with enough accuracy for conceptual design. Future plans for RAGE development, especially in transition to and from CAD, will further improve its applicability to conceptual aircraft design.

Acknowledgments

The authors would like to acknowledge some colleagues at Desktop Aeronautics, Inc., who have helped in the development of RAGE. Peter Sturdza contributed to several algorithms and to the programming several parts of RAGE. Vianney Maniere contributed in the development of a finite-element, structural analysis model using the RAGE libraries. The authors would also like to acknowledge all other colleagues who have contributed to RAGE development in any way. Finally, the authors acknowledge NASA Langley and Chau Chang for providing funding to continue the development of RAGE over the past 3 years under SBIR contract NNL08AA08C.

References

1. Hahn, A., “Vehicle Sketch Pad: A Parametric Geometry Modeler for Conceptual Aircraft Design,” AIAA 2010-657, Jan. 2010.
2. “AVID LLC - AVID PAGE,” URL: http://www.avidaerospace.com [cited January 2010].
3. Cramer, E. J. and Gablonsky, J. M., “Effective Parallel Optimization of Complex Computer Simulations,” AIAA 2004-4461, August 2004.
4. Bowcutt, K., “ A Perspective on the Future of Aerospace Vehicle Design,” AIAA 2003-6957, December 2003.
5. Rodriguez, D.L. and Sturdza, P., “A Rapid Geometry Engine for Preliminary Aircraft Design,” AIAA-2006-0929, January 2006.
6. Sturdza, P, "Extensive Supersonic Natural Laminar Flow on the Aerion Business Jet," AIAA 2007-0685, January 2007.
7. PASS, Program for Aircraft Synthesis Studies, Software Package, Ver. 1.7, Desktop Aeronautics, Inc., Palo Alto, CA, 2005.
8. Allison, E., Kroo, I.M., Sturdza, P., Suzuki, Y., Martins-Rivas, H., “Aircraft Conceptual Design with Natural Laminar Flow,” 27th International Congress of the Aeronautical Sciences, Sep. 2010.
9. Aftosmis, M. J., Berger, M.J., and Adomavicius, G., “A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries,” AIAA 2000-0808, January 2000.
10. “Icon Aircraft: Sport Flying Revolution,” URL: http://www.iconaircraft.com, cited Dec. 2010.
11. “Blender Home,” URL: http://www.blender.org, cited Dec. 2010.
12. “Octane Renderer,” URL: http://www.refractivesoftware.com/, cited Dec. 2010.
13. “GridTool: A Surface Modeling and Grid Generation Tool,” URL: http://geolab.larc.nasa.gov/GridTool/, cited Dec. 2010.
14. “Open CASCADE Technology, 3D modeling and numerical simulation,” URL: http://www.opencascade.org/, cited Dec. 2010.