2011

A case study in CAD design automation

Andrew G. Lowe

Nathan W. Hartman

A Case Study in CAD Design Automation

Andrew G. Lowe and Nathan W. Hartman

Abstract

Computer-aided design (CAD) software and other product life-cycle management (PLM) tools have become ubiquitous in industry during the past 20 years. Over this time they have continuously evolved, becoming programs with enormous capabilities, but the companies that use them have not evolved their design practices at the same rate. Due to the constant pressure of bringing new products to market, commercial businesses are not able to dedicate the resources necessary to tap into the more advanced capabilities of their design tools that have the potential to significantly reduce both time-to-market and quality of their products. Taking advantage of these advanced capabilities would require little time and out-of-pocket expense, since the companies already own the licenses to the software. This article details the work of a small research team working in conjunction with a major turbine engine manufacturer endeavoring to make better use of the underutilized capabilities of their design software. By using the scripting language built into their CAD package for design automation, knowledge-based engineering applications, and efficient movement of data between design packages, the company was able to significantly reduce design time for turbine design, increase the number of feasible design iterations, increase benefits from relational modeling techniques, and increase the overall quality of their design processes.

The design of turbine engines involves creating, modeling, and documenting the development of airfoil geometry for turbine, impeller, and compressor blades. This process is highly iterative due to the circular revisions made between design and analysis groups chasing the optimal airfoil shape and performance. Airfoil blades are a crucial component within a turbine engine, and their design covers many engineering disciplines such as thermodynamics and statics. For both analysis and manufacturing, these airfoils are modeled in a CAD system. However, the complex shapes of airfoils make this difficult. They are typically modeled using b-splines or NURBS, and the development of methods to do this has been ongoing for decades (Corral, Roque, Pastor, & Guillermo, 2004; Korakianitis & Pantazopoulos, 1993)). After revisions are made, geometric data are often re-engineered and recreated within the CAD system. This process ranges from hours to days because the current methods of creating the airfoil models in the CAD system are not parametric, (i.e., the geometry is not associated with the engineering definition of the airfoil after the model is created). A turbine engine can contain as many as of 20 different airfoils, so any improvement in the time for one design iteration will have a beneficial effect on the total design process. In addition, additional benefits can be realized depending on whether a turbine, compressor, or fan blade is being designed, as the geometric complexity of each part varies from relatively simple to highly complex.

According to O'Brien et al. (2006), the knowledge-based engineering (KBE) techniques can make a substantial impact in the design of engineering products. It is gaining prominence as a major tool to speed up product development by capturing knowledge from engineers and designers and embedding that into software configuration (Bermell & Fan, 2002; Prasad, 2005; Rosenfeld, 1995). This knowledge is then used to assist designers while they create products within the CAD system (Hunter, Rios, Perez, & Vizan, 2005). KBE systems are used to automatically create objects (Clark, 2001; Sekiya, Tsumaya, & Tomiyama, 1998), assist designers while they create objects (Carleton, 2005), and compare the cost versus efficiency of created objects (Susca, Mandorli, & Rizzi, 2000).

The industrial research partner in this project does its CAD design in Siemens PLM NX and ports their models into assorted versions of ANSYS and various other in-house applications for analysis. At the beginning of the project the design process was almost totally manual – aerodynamics engineers would pass point cloud data representing turbine airfoils to modelers who would spend one or more full workdays constructing a CAD model from the data. This time encompasses only the airfoil itself and not any of the turbine wheel attachment points or internal cooling geometry. There were no standards in place, so each modeler created their airfoils in their own way which complicated and unnecessarily extended the time required for design changes and additional design iterations. Altering a turbine model would either require adding the task to the queue of the original modeler, each of whom works on several projects concurrently, or enlisting an available modeler to decipher the original modelers techniques and make the necessary adjustments. Even in ideal circumstances, the time to make a design change would be roughly equal to the time required to make the original model.

The company was interested in the capabilities of Knowledge Fusion (KF), a scripting language built into the NX CAD package to automate, standardize, and streamline this process. It had several objectives in mind for the prospective KF application. The first was to reduce the overall design process time by automating the repeated tasks involved in creating the initial airfoil CAD model from the aero engineer's point cloud data. The second was to reduce the time required for design changes and design iterations. By building on objectives one and two, it hoped to standardize the process, both to reduce the likelihood of costly errors in the existing fully manual process and to have consistent models suited to more efficient or automated importation and meshing in analysis software.

Initial requirements were for a KF application capable of reading the raw point cloud data provided by the aero engineers and automatically generating a solid model to which a modeler could add the necessary geometry for attachment points and internal cooling. Ideally the application would be user friendly enough that the aero engineers, who have no CAD training, would be able to generate the initial airfoil model themselves and verify that the solid model conforms to their design intent before passing the model off for final modeling, analysis, and production, a capability they did not currently have. If the application proved robust in generating the initial airfoil solid, additional capability would be added to the application allowing for automation of additional features, including framework for internal cooling geometry, representations of thermal coatings, and NX-specific settings to conform to company design policy and to make the final modeler's job easier.

Project Background

Knowledge Fusion (KF) is a procedural, object-oriented scripting language built into the NX CAD package. Generic Windows-style menus and dialog boxes can be created and tied into KF applications with UI Styler, a user interface design tool also built into NX. KF applications run from simple text files, so they do not need to be compiled on each computer they are to be used with. This makes distribution of the applications throughout a corporation a simpler matter, and it also gives a company the ability to store the application files on a server to which employees can point their copy of NX and run the application without having to download the files.

KF offers most of the basic capabilities one would expect from a programming language – conditional logic, looping, file input/output, basic math, text parsing and string manipulation. The language's vast function library allows the user to call virtually every action available in NX's traditional graphical user interface. With basic programming architecture and the large library of geometry-related functions, a KF application can create automatically almost any model a trained human could design by hand (Golkar, 2006).

Program Capabilities

The automated turbine design application was developed in stages by a series of small research teams and individuals, each building upon the work of the previous researchers and adding features as each stage was determined to be robust enough for production. The initial application would only read in the point cloud data and create the solid model, but through succeeding iterations all desired capability was added and determined to be stable.

Solid Model Generation

The company for which the application was designed uses a handful of proprietary file formats for their turbine point cloud data depending upon the application used to design the turbine and the location at which it was designed. Each format is roughly similar regarding the way the points are organized. The airfoil points are divided into sections, each laterally ringing the airfoil. Some formats use a fixed number of sections, others support a dynamic number. Three separate parsing functions were developed to read in the data and store them in a consistent manner to avoid costly and inefficient repetitions of modeling functions within the body of the program.

The user interface requires the user to select the appropriate file format before the data is fed to the program. At that point the parser ignores any existing header data, then reads and stores the points in a three-dimensional array (referred to as lists in KF), the top-level array holding an array of points for each section. The application then loops through the list, drawing a spline through each array of section points. Each spline is stored as an element in a new array, which is, in turn, looped through by one of NX's multi-section solid operations using each spline as a guide to create the airfoil solid. Due to the complex curvature of turbine airfoils and the necessity of absolute smoothness and precision in the model, several multi-section solid functions had to be evaluated by the research team and the company's modelers before an appropriately precise and robust operation was found (Farin, 1997).

Layering and Coloring

The partner corporation has strict modeling guidelines regarding the development of models. Each type of reference and final geometry has to be placed on a different layer in NX both to avoid graphical cluttering of the final model and to make modifications and design changes easier as the part file circulates through different modelers during its design. For the airfoil generation application to be useful in a production context, the models it creates must conform to these standards. When every point, line, surface or solid is created, the application puts it on the appropriate layer. There is a set of default layers built into the program, but these can be changed before model generation through the user interface.

Due to the sheer volume of geometric data that the application creates, it was deemed necessary to alleviate potential visual clutter by making each piece of geometry noticeably different from the rest. The same command that allows for specific layer placement of newly generated geometry also allows the color to be controlled. Similar to the layering capability, default colors are stored for each piece of geometry, but these can be changed through the user interface. The layered geometry also reduces visual clutter, since the user can quickly hide construction or other geometry without being familiar with the feature tree generated by the application.

Face Tagging and Finite Element Analysis (FEA) Integration

The partner company is developing an automated CAD/FEA integration, discussed in detail in the BENEFITS section, that relies on named or "tagged" faces for automated meshing. When the initial solid model is generated, the operation is repeated with the added specification that the generated geometry be a surface instead of a solid, effectively wrapping the solid airfoil in a geometrically identical blanket. The hub and tip surfaces are then cut away from the initial surface wrap using the uppermost and lowermost section splines. Each point cloud file type has the potential to describe each section with a differing number of points, so the points in each section that represent the border between the four vertical faces must be identified based on the type of file in which they are contained. Once identified, the border points are saved in arrays where the program loops through them to create cutting splines running from the hub to tip of the airfoil. The surface wrapping the airfoil vertically is split into four individual faces using these splines as references. Each face is named using a convention recognized by the automated meshing program.

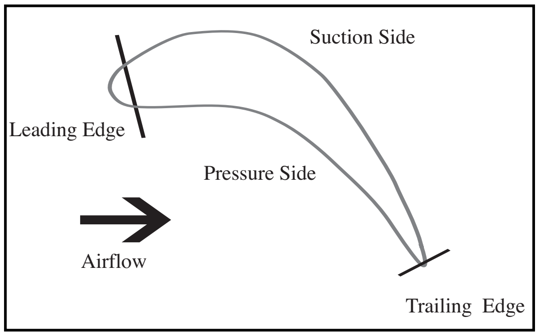

Figure 1: The key vertical faces of a turbine airfoil.

Internal and Coating Geometry

The turbine portion of an engine operates at very high temperatures, often exceeding the melting point of the metal of the airfoils, so some turbine airfoil designs include hollow internal geometry for cooling or a spray-on coating of a thermal-resistant compound (Newman, 2002). The coating can add weight and thickness that affects results of mechanical and aerodynamic analysis, and the internal cooling geometry is often complex, requiring significant time to model. Both could benefit from automation.

After the initial airfoil solid is created, the application's user interface can be reopened and the user can select whether the blade will be solid, hollow, coated, uncoated, or any combination thereof. If a solid or coated blade type is selected, the user is prompted to load a text file, referred to as a wall file, which contains thickness data for the selected operation. As shown in Figure 1, every wall file has offset information for both the coating and cooling geometry grouped by section and face (leading edge, suction side, trailing edge, pressure side). Each section is grouped by face, and each group of face data contains several pairs of numbers; one for the distance the coating or cooling geometry is offset from the original sections, and one that defines where on the face the offset will be located, expressed as a percentage of the face's total length.

To create the coating geometry, the program loops through each section spline of the original solid, and then through each face of each section. For each face, the program loops through the coating data in the wall file and offsets points outward from the original spline based on the thickness data provided; it then stores the offset points in an array. Once all the points are stored, they are looped through, section-by-section, and a new spline is drawn through them. The splines then are used to create a solid that represents the coating.

Creation of the cooling geometry operates on very similar principles, but with some added complexity. The coating offset thickness is by nature uniform, but the offsets for internal geometry are variable to allow for tailoring of the cooling properties as well as the physical strength of the resultant hollow blade. During initial design, a solid would be created from the cooling offset splines to hollow out the airfoil solid, but it was found in testing that modelers could finalize the complex internal geometry faster without the solid or a hollow blade, just using the cooling offset splines as references. The coating splines are visible in green outside the airfoil solid. The blue splines represent the hollow core as shown in Figure 2.

Gas Path Representation

All geometry in a turbine engine will at some level reference the path air takes through the compressors, into the combustion chamber, and out through the turbines. This is referred to as the gas path, and the splines that represent it would essentially be the highest level skeleton model for a completely relationally modeled

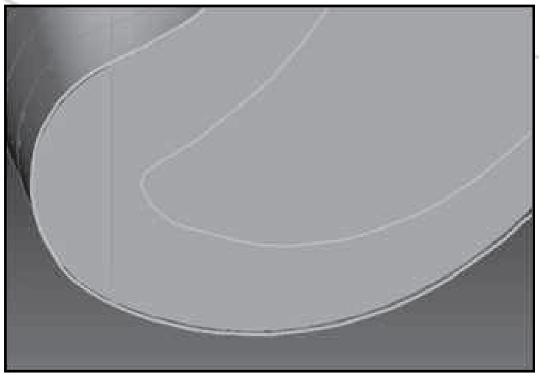

Figure 2: The leading edge of a hollow coated airfoil model done by the KF application. Curves digitally enhanced to aid visibility.

turbine engine. A tool for generating the gas path inside the turbine model was built into the airfoil generation application. The users may select whether they wish to display the hub annulus, tip annulus, or both through the user interface, and they are then prompted to load a text file. The text file contains a series of simple x,y,z points (typically 100-150) that the program loops through to create a spline.

Benefits

Time Savings

The first and most obvious benefit of automating a process is reduced time for executing that process. Surveys indicated that the typical modeler would take from 5 to 8 hours to create a solid model from Computational Fluid Dynamics (CFD) point cloud data. Using the automated airfoil generation tool, modelers were able to create the same airfoil in a uniform and consistent manner in 4 to 6 minutes. Members of the research team with more familiarity with the application would routinely generate airfoils in two minutes or less. The company estimates use of the KF tool will save approximately three quarters of a million dollars in direct costs alone on a single engine project.

The time savings created by the tool extend farther through the design process than the creation of the initial turbine model. Before the automated airfoil generation tool was put into production use, any change to an airfoil design would encounter a bottleneck in the modeling department. Each modeler works on multiple projects, and multiple parts per project, so a design change would have to be queued into the original modeler's current tasks, which would create a period of up to two weeks between a change in design from engineering and analysis where nothing could be done with the design. The dead time could be reduced by finding a modeler with available time to make a design change, but that modeler would have to waste considerable time deciphering the original modeler's techniques to make the appropriate changes. Even if the original modeler were immediately free to execute design changes, if the change was in the original CFD definition, the airfoil the model would have to be rebuilt virtually from the ground up. Not only would the modeler have to reinvest the 5 to 8 hours required to make the initial airfoil solid, but the additional time necessary to re-model attached geometry dependant on the airfoil, such as internal cooling slots or external features for mounting the blade to a turbine wheel.

By automating the process and making airfoil models the same way every time, any modeler can update the model just as efficiently as the modeler that originally created it. The tool requires no knowledge of CAD to operate, so a modeler would not be required to update the model provided robust relational modeling techniques were used to create cooling and turbine wheel attachment geometry. Using the application, an engineer could run analyses on several different design iterations in a single day without having to utilize a modeler for each design change.

Increased Iterations

Another benefit for reducing process execution time is the ability to run more design iterations in equal or less time for equal or less cost. As stated previously, the airfoil generation application was designed so the KF code would stay embedded with the part file after the application was run, allowing the user to re-run the program at any time, even after more geometry had been added, either by re-selecting the application though the NX menus, or by double-clicking on any of the geometry it creates. When the application is reopened, the user may replace any of the text files that define the automatically created geometry and the application will update all relevant geometry using the new text files. Because the geometry is updated and not deleted and recreated, any geometry added to the model that relationally references the application's geometry will also update.

There is much more than just an airfoil in a final turbine blade model. The initial airfoil model is purposely made longer on both the hub and tip ends to ensure that the part of the airfoil that will be used has contiguous curvature, so both ends must be trimmed. Complex internal cooling geometry is added when necessary, and geometry defining the blade's attachment point to the turbine wheel (often referred to as a "fir tree" because of its uncanny resemblance to a profile view of a Christmas tree) must be added, as well. Since it is not feasible to manually redefine the many thousands of points that define the CFD definition of the airfoil, a modeler would have to start from scratch, creating a new initial airfoil and rebuilding all the aforementioned geometry with each new design iteration created. With the airfoil generation KF application, that time investment is still required for the initial iteration, but subsequent iterations require only that the appropriate text files be changed, reducing the time required for additional iterations from days or weeks to a matter of minutes.

Robust relational solid modeling techniques are key for this process to work in a production context. Any piece of geometry not relationally referencing either the KF application geometry or another piece of geometry that relationally references it will not update with the rest of the model and could take longer to fix that it would have to create it from scratch. Because of geometrically complex nature of turbine blades, the type of relational referencing must be tested for robustness on updating. Several of the company's modelers were tasked with developing a standardized, documented, and robust method of modeling the additional geometry. At their request, a handful of axial and radial splines were added to the application to make fully relational modeling easier. Using the modeling techniques the partner company developed, final, fully modeled turbine blades can be completely controlled by the KF application's text files.

Increased Process Control and Quality

Turbine blades have a high degree of geometrical complexity and require skilled modelers to model them effectively. In the past, once aero engineers finalized the design of their airfoil in a CFD program, they were forced to pass their point cloud file to a modeler to create the CAD file, who would in turn pass the model on to analysts for FEA and so on until production. It was assumed that the CAD model would conform to the aero engineer's design intent, but there was no process in place to establish this empirically. Since the multi-section solid operation that creates the airfoil model uses the point cloud section as a reference, the model would be valid at those points, but the validity of the surfaces between the sections was in question.

The airfoil generation KF application removes virtually all the modeling skill required to create the initial solid. By distributing the application with a simple two-page user guide to aero engineers, the engineers were able to create the initial airfoils themselves. A separate KF application referred to as the Point-Body Comparison tool (PBC) was distributed as well. The PBC accepts a point cloud text file in the same format that the airfoil generation tool uses, then prompts the user to identify the pressure side, suction side, leading edge and trailing edge faces on the airfoil solid. When the application runs, it creates each point, measures the distance between it and the appropriate face, then colors the point based on its distance – ranging from blue, representing little or no difference, to red, representing larger differences. The result is an easy-to-interpret graphical representation of the solid model's conformity to original CFD design intent. By using the airfoil generation tool to create the initial airfoil, then exporting a new point cloud file representing the same airfoil but with sections in different places, now an aero engineer can verify that an airfoil model conforms to the original design intent before it ever leaves their control.

Quality issues surrounding that transfer of data between departments and employees also arose when the company was using an all-manual modeling process. Due to a lack of process documentation and the variety of dissimilar modeling techniques, it was not uncommon for necessary operations such as movements in the coordinate system or flipping of models to be done by one employee, then passed to another employee who would perform the operation again, sometimes resulting in costly errors. The airfoil generation application and the development of the relational modeling techniques that accompanied it not only standardized the modeling process, but also made it much easier for the company to rigidly define the roles of each employee in the design process, always performing each modeling operation at the proper time, always performing each translation and flip at the proper time.

Efficient Analysis Integration

Analysis is just as important to an effective product as the initial model itself, and like modeling, creating a robust mesh for FEA can be just as manually labor-intensive as creating CAD geometry. Just as each CAD modeler tends to employ a unique technique for creating a model, analysts tend to create meshes in their own way, which can lead to small differences in the final output. To both remedy this potential problem and to speed the design cycle, the partner company is interested in developing a method for automated meshing. Its method relies on indentifying four key vertical surfaces and two key horizontal surfaces on the airfoil to be used as references in the meshing operation. The vertical surfaces are the pressure side, suction side, leading edge, and trailing edge. The horizontal surfaces are the hub and tip.

The combination of the airfoil generation application's face tagging plus fast, text-file based design iterations and the company's development of an automated meshing tool makes for an incredibly fast analysis and optimization process. An analyst can sit down with a variety of CFD point clouds or wall file data or both, make a model for each desired combination, mesh and analyze them in batch and interpret the resultant data. The only factor significantly limiting the number of designs they can analyze in a day is the speed of the computer running the FEA program. The company estimates that the combination of the KF application and FEA integration will save approximately $3.7 million in direct costs on a single engine project.

Conclusion

Through thorough testing and evaluation, automated CAD design via built-in scripting tools has proven to be an effective way of reducing design time, increasing the number of feasible design iterations, increasing the quality of processes and the company's control over them, and enhancing integration with other automated processes outside of CAD. The turbine engine manufacturer has deemed the KF application robust and reliable and has recently put it into production on a current engine project.

Such automation also opens the door for further enhancements to the design process. Development of similar applications is possible for most engine components, and could automate most modeling required in an engine project. By using text files to control important engineered data in tandem with robust relational modeling techniques, it would be possible to achieve the company's goal of total gas path design and engine reuse. By total use of relational modeling with the gas path as the highest level reference, an existing engine model could be reused on a new project. The gas path could be changed to alter the overall sizes and airflow, and text files controlling airfoils and combustor geometry could be changed to meet new thrust and efficiency requirements. This, combined with quick iterations and automated analysis time-to-market, has the potential to drastically reduce overall time-to-market.

Andrew Lowe is a graduate student at the Purdue University Department of Computer Graphics Technology. He has worked with a variety of industry partners doing research into the integration of Virtual Reality, automation, and emerging technologies with CAD and the design process.

Nathan W. Hartman is an Associate Professor and Assistant Department Head in the Department of Computer Graphics Technology at Purdue University, West Lafayette, Indiana. He is a member of the Gamma Rho Chapter of Epsilon Pi Tau.

References

Bermell, P., & Fan, I. (2002). *A KBE System for the Design of Wind Tunnel Models Using Reusable Knowledge Components.* Paper presented at the VI International Congress on Project Engineering. Barcelona, Spain.

Carleton, S. (2005). *Design rationale in a knowledge-based computer-aided design environment.* Unpublished master's thesis. Purdue University, West Lafayette, IN.

Clark, A. L. (2001). A solid modeling services architecture for KBE applications. *ACM Symposium on Solid Modeling and Applications: Proceedings of the sixth ACM symposium on Solid Modeling and Applications.* Retrieved from ACM Portal database.

Corral, Roque, Pastor, & Guillermo. (2004). Parametric design of turbomachinery airfoils using highly differentiable splines. *Journal of Propulsion and Power, Vol. 20*(2), 335-343.

Farin, G. E. (1997). *Curves and surfaces for computer aided geometric design.* San Diego, CA: Academic Press.

Golkar, M. (2006). *Development of Knowledge-Based Engineering Support for Design and Analysis of Car Components Using UGS NX-Knowledge Fusion.* Lulea University of Technology.

Hunter, R., Rios, J., Perez, J. M., & Vizan, A. (2005). A functional approach for the formalization of the fixture design process. *International Journal of Machine Tools & Manufacture, 46*, 683-697.

Korakianitis, T., & Pantazopoulos G. I. (1993). Improved turbine-blade design techniques using 4th-order parametric-spline segments. *Computer-Aided Design, Vol. 25*(5), 289-299.

Newman, D. (2002). *Interactive aerospace engineering and design, aircraft propulsion,* New York: McGraw-Hill.

O'Brien, W., et al. (2006). Using knowledge-based solid modeling techniques and airfoil design data: A case study in developing an airfoil seed part generator. *Proceedings of The 2006 IJME - INTERTECH Conference, Union, NJ, October 19 – 21, 2006.*

Park, J., Park, S., Hwang, I., Moon, J., Yoon, & Y., Kim, S. (2004). *Optimal blade system design of a new concept VTOL vehicle using the Departmental Computing Grid system.* School of Aerospace and Mechanical Engineering, Seoul National University, Seoul, Korea.

Phillips, R. E. (1997). Dynamic objects for engineering automation. *Communications to the ACM, 40*(5). Retrieved from ACM Portal database.

Prasad, B. (2005). What Distinguishes KBE from Automation. *COE NewsNet.* Retrieved from http://www.coe.org/newsnet/Jun05/knowledge.cfm#1.

Rosenfeld, L. (1995). Solid modeling and knowledge-based engineering. In D. LaCourse (Eds.), *Handbook of solid modeling* (pp. 9.1-9.11). New York: McGraw-Hill.

Sekiya, T., Tsumaya, A., & Tomiyama, T. (1998). Classification of knowledge for generating engineering models: A case study of model generation in finite element analysis. Research in Artifacts, Center for Engineering, University of Tokyo, Japan.

Spitz, W., Golaszewski, R., Berardino, & F., Johnson, J. (2001). Development cycle time simulation for civil aircraft. *NASA Langley Technical Report Server.* Retrieved from ACM Portal database.

Susca, L., Mandorli, & L., Rizzi, C. (2000). *How to represent intelligent components in a product model.* Department of Industrial Engineering, University of Parma, Italy. Department of Mechanical Engineering, University of Ancona, Italy.