Development of Turbine Blade Generator Using UGS NX 2.0

by

Andrew Hudecki
ahudecki@purdue.edu
Computer Graphics Technology
Purdue University

Nathan W. Hartman, Ed.D.
nhartman@purdue.edu
Computer Graphics Technology
Purdue University

Joseph Miller
millerjs@purdue.edu
Computer Graphics Technology
Purdue University

Mark Coster
mcoster@purdue.edu
Computer Graphics Technology
Purdue University

Kirk Layman
klayman@purdue.edu
Computer Graphics Technology
Purdue University

Abstract

This paper details the process of developing a turbine blade generator that incorporates design methodology to the specifications of a leading aerospace company. The project encompasses the research and development of generating turbine blade geometry using UGS NX 2.0, Knowledge Fusion, and UI Styler. A milestone for the aerospace manufacturer is to reduce the design and engineering timeframe to produce a turbine engine, and in doing so, several practices need to be automated to reduce production time. A group of Computer Graphics Technology (CGT) students from Purdue University collaborated with a leading aerospace company to determine if automation of turbine blade geometry was possible in the UGS NX 2.0 environment using Knowledge Fusion. The students' main goal was to justify the time savings of the automated turbine blades within the computer-aided design (CAD) system and document the limitations of the UGS NX 2.0 release. The following paper will document the process of the automation of turbine blade geometry.

I. INTRODUCTION

Traditionally utilized methods of creating turbine blade geometry in CAD systems can be lengthy and prone to human error. This process can range from hours to days using traditional 3D modeling techniques. Previous methods required a modeler to construct a turbine blade from a previously created airfoil generation tool. The modeler would then add the necessary hollow cavities within the airfoil to create a proper cooling mechanism for the turbine blade.

Turbine blades act as powering mechanisms for various components of the engine. As air is pulled into the engine, it is converted to high pressure by compressor blades. The air is then ignited in the combustion area, where fuel is mixed with the pressurized air to produce a high-pressure, high-velocity gas. The ignition of this gas moves through the turbine blades, which are facing the opposite direction of compressor blades, thus providing forward thrust by turning the turbine blades [1]. (See Figure 1)

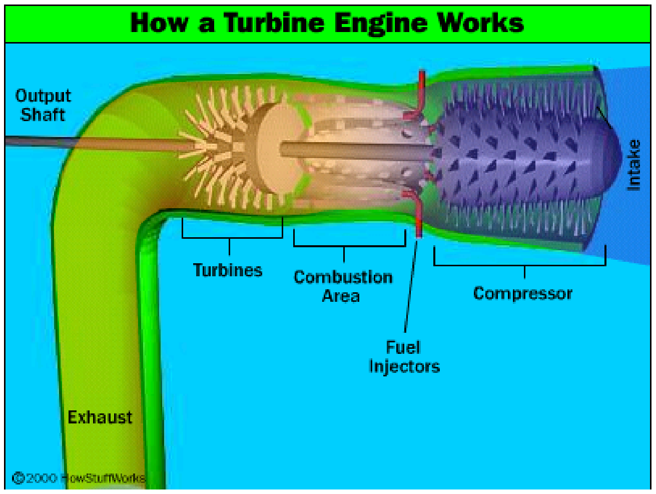


Figure 1: Basic Turbine Engine
Figure Source: http://www.howstuffworks.com/turbine3.htm

Turbine blades use a different airfoil definition than that of compressor blades. Turbine blades differ in the respect that they require cooled air to be pumped through the blade internally so the ignited gas does not cause the composite metal to fail.

Previous Project Background

The former group team that worked on this project developed an Airfoil Seed Generator with a focus on compressor blades during the Spring 2006 semester. The project discovered that using Knowledge-Based tools enhances the automation of solid model compressor blade geometry and effectively saves time during the creation process [2]. The current Purdue University project (underway in 2008) with the aerospace manufacturer will expand upon this Knowledge-Based tool in order to incorporate the automation of turbine blades.

Current Project Overview

The Spring 2007 Semester project scope was to be a key element in the scheme of bringing a new product to market in a 24-month time frame utilizing a Knowledge-Based Engineering solution. The project team's role was to determine if the use of Knowledge-Based Engineering practices, through the use of Knowledge Fusion in UGS NX 2.0 is a time saving means of obtaining this goal.

Knowledge-Based Engineering (KBE) is a method of capturing knowledge in a CAD environment through the practice of geometry reuse [3]. KBE establishes modeling protocols that if used properly, can save time for the modeler through the elimination of repetitive geometry creation. Once a KBE system has been developed, new users can access and modify different parameters to create the desired geometry.

The project involved working with UGS's NX 2.0 CAD tool and Knowledge Fusion to automate the creation of turbine blades. Knowledge Fusion is an object-oriented, hierarchical, declarative scripting language that works directly within UGS NX 2.0 [4]. Knowledge Fusion relates geometric features to different classes within the scripting language, and allows for dynamic model creation. Knowledge Fusion allows for the development of custom applications for UGS, as it provides the ability to make modifications to the product without needing to recreate previously defined geometry.

In conjunction with Knowledge Fusion, a user interface was designed with UI Styler, a module that can be launched within UGS NX 2.0. (See Figure 2 below for main Turbine Blade Generator Interface.) UI Styler creates a user-friendly interface that works directly with the Knowledge Fusion code as its backbone. Upon the completion of the Knowledge Fusion code, the solution was validated through the comparison of traditional and automated modeling methods comparing average mean times.

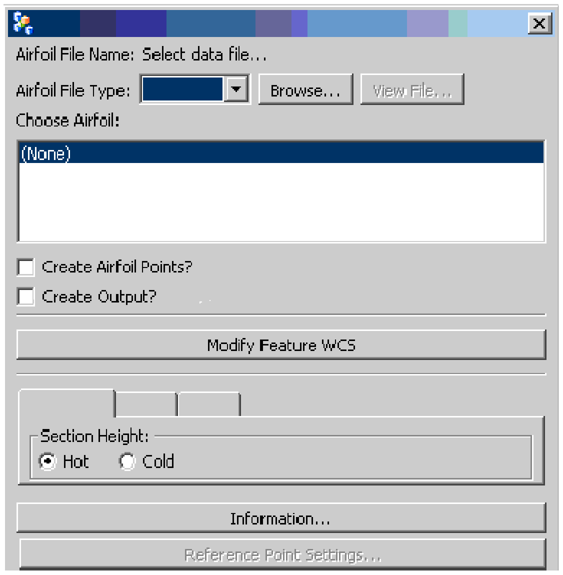

Airfoil File Name: Select data file...
Airfoil File Type:
Choose Airfoil:
(None)

Figure 2: Main Turbine Blade Generator GUI

II. BACKGROUND LITERATURE

Initial research for the Turbine Blade Generator project covered several different topics. Primary research focused on the KBE aspect of the project. Secondary research involved Knowledge Fusion, Parametric Modeling, and the traditional methods of turbine blade geometry creation. Tertiary research components focused on user interface testing.

"Knowledge Intense CAD systems consist of commonly accessible knowledge sources, which can be applied to relevant problems, where a knowledge source consists of suitably structured knowledge to tackle a specific problem" [5]. Many sources of research indicate that to remain competitive in a research and design market, different aspects of the engineering process need to be automated. This allows companies to reduce production time and "bring product design into closer harmony with manufacturing, support, finance, procurement, sales, marketing, and other domains" [6]. KBE captures design intent as well as engineering methodology and have significant reduction in labor for the creation of future products. Research conducted on KBE verified previously known material about KBE and eliminated any confusion of the functionality and limitations for what KBE systems can accomplish.

Knowledge-Based Engineering (KBE) "has gradually gained prominence as a major tool to speed up product development…" [7]. KBE captures knowledge form engineers and designers and embeds that within a particular program – in this case, a CAD system [7, 8, 9, 10]. Designers are then able to create 3D models within the system in a more efficient fashion [11]. KBE systems are capable of automatically creating objects [12, 13], assisting designers while they create objects [14], and comparing the cost versus efficiency of created objects [15, 16]. There are many examples of companies that successfully used KBE systems for the development of their products. For example, Park et al. [16] developed a method that produced multiple variations of blade designs for a vertical take-off and landing aircraft, performed finite element analyses, and then chose the optimal blade based on the results. A primary reason for the implementation of KBE systems around the world is the reduction of design and analysis time [17]. In this particular case study, parametric modeling functionality and creation of turbine blades, and the use of the Knowledge Fusion application itself came together in order to create the KBE application.

Turbine blade geometry specifications, particular to the aerospace manufacturer, were provided to the students. This allowed the design intent of the turbine blade geometry to be understood. The specific content of this documentation contained proprietary data that could not be discussed in detail in this research paper. Within the documentation given to the project group, the aerospace company's procedures of defining turbine blade geometry were addressed. Here, steps needed in order to validate the resulting product were described. These specifications were incorporated into all phases of the development of the Turbine Blade Generator.

Tertiary research included studying methods of how to extract meaningful information from administered surveys. Such examples included using a Likert scale in order to identify the user's level of understanding and ease of use of the product. The examples also addressed the need for a comment area, where the user can add suggestions or additional comments [18].

III. PROJECT DEFINITION

During the initial meeting with the aerospace engine manufacturer in February 2007, the primary goals of the project were discussed and ranked in order of priority. Fellow attendees included engineers and designers with experience in turbine geometry creation, thermodynamics, and stress analysis. Upon the completion of the meeting the priority list was concluded as followed:

Priority List

1. Improve tip and hub surface definition of the Airfoil Seed Generator
    - Create single continuous face
2. Core Definition Tool
    - Gill slot definition based off wall thickness
3. Mass Properties Generator
    - Set boundary planes for section analysis

IV. DEVELOPMENT OF APPLICATION

Priority 1 - Fix tip and hub in the Airfoil Seed Generator Overview

In the development of the first priority, the team had the task of addressing the problem of the current tip and hub definition. Working with the leading aerospace manufacturer, it was deemed necessary that a turbine blade have a continuous face for the tip and hub. Unlike turbine blades, compressor blades require a multi-face tip and hub for meshing purposes in respect for the much smaller geometry at the leading and trailing edges. View Figure 3 for areas addressed.

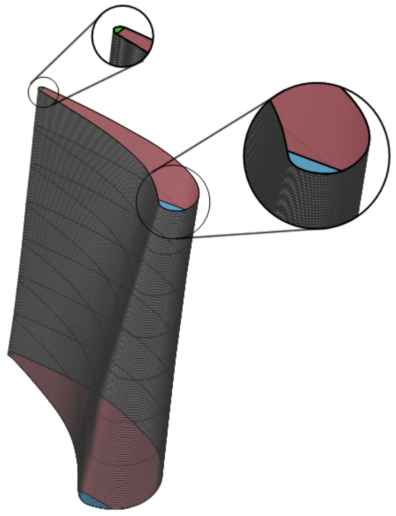


Figure 3: Illustrating the Tip and Hub faces addressed in Priority 1

Limitations found in Creating a Continuous Tip and Hub of the Turbine Blade

In addressing the Turbine Blade Generator, one limitation was found. In UGS NX 2.0, the "bounded plane" command can be used manually. However, no functionality to utilize this command in Knowledge Fusion exists.

The next possible solution for fixing the tip and hub utilized the "Through Curve Mesh" command. This tool provided the result needed to complete the task of creating a continuous face for the tip and hub. The "Through Curve Mesh" command works by selecting the pressure and suction side edges first (longer splines) and then selecting the leading and trailing edges second. The CAD system proceeds to generate an appropriate surface between the edges.

Problems with the "Through Curve Mesh" command occurred after the UGS NX 2.0 software was exited. Even if a successful execution of the code was performed, the following error occurred upon a retest of the same Knowledge Fusion code:

```
Unable to accomplish the operation due to following error encountered during update:

Internal error: memory access violation O/S ERROR: signal 11, code 00000000

Unable to build/execute the styler dialogue.
```

Upon investigation of this problem, it was discovered that the version of UGS NX 2.0 at the leading aerospace manufacturer differed from that of Purdue University. After Purdue University upgraded to the release of UGS NX 2.0.6.2 in use at the leading aerospace company, the problem was resolved. This issue was overlooked at the early stages of the project because differing software versions were not an issue for the previous CGT project. In further understanding this obstacle later, the error message was found to be a generic runtime error found in Knowledge Fusion. The error still occurred periodically throughout the project, but did not have the same impact prior to the UGS NX 2.0 upgrade. On the assumption that the impact to the future of the project would be minimal, it was decided that an instability issue exists within UGS NX 2.0 that tends to create the described error.

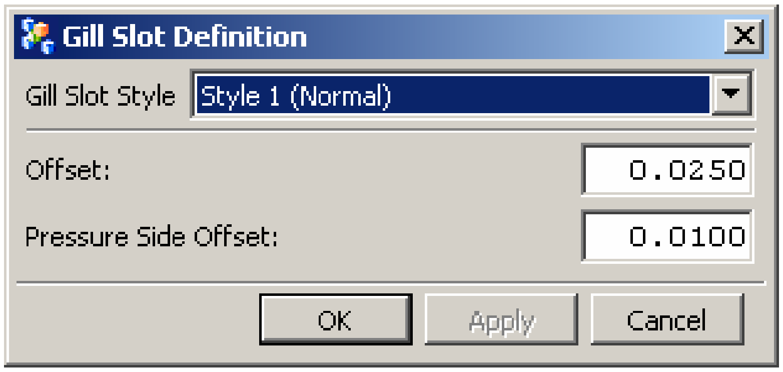
Within the graphical user interface (GUI), the possibility of utilizing an IF/ELSE statement to call proper functions was explored. The IF/ELSE statement allowed either a compressor blade or turbine blade to be generated within the same interface. One of the limitations of Knowledge Fusion within UGS NX 2.0 is that once an airfoil definition file is loaded, the software refuses to execute the same application for a different function during the same session of the software. The user would then have to exit out of UGS NX 2.0 entirely and restart the program.

The solution was to create two separate .dfa files for the airfoil creation instead of using an IF/ELSE statement. Within the Knowledge Fusion code, a different script is loaded when the user generates both a compressor blade and a turbine blade during the same session.

Priority 2 - Core Definition Tool Development Overview

The turbine blade itself has a tendency to reach temperatures − exceeding − 3,000 − degrees Fahrenheit at the outer surface. In order to combat the performance hindrance that results from these high temperatures, the aerospace manufacturer applies what is known as a “gill slot” to the trailing edge of the hollow turbine blades. This slot provides the necessary means for the cooling air to exit the core of the turbine blade. By utilizing gill slots, the aerospace manufacturer is able to improve the performance and longevity of their turbine blades.

Many challenges were faced by the team throughout the process of developing functionality within Knowledge Fusion to automatically generate a gill slot for the turbine blades. In order for the gill slot geometry to be generated appropriately, the curves that compose the slot had to follow the contours of the complex outer turbine blade geometry. In essence, the arcs, lines, and curves that were utilized in the creation of a gill slot on one cross-section of the turbine blade changed slightly at the next cross-section. This added a level of complexity to the Knowledge Fusion coding task. Once completed, the gill slot curves were appropriately generated, creating a protrusion from the pressure side wall at the trailing edge of the turbine blade. GUI and solution are shown below in Figures 4 and 5.


Gill Slot Definition

Gill Slot Style: Style 1 (Normal)

Offset: 0.0250

Pressure Side Offset: 0.0100

Figure 4: Turbine GUI for Gill Slots

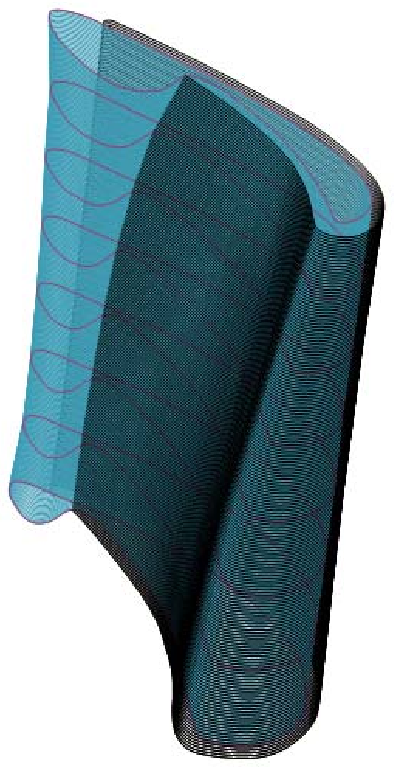


Figure 5: Illustrating the Turbine with Gill Slots

Delimitations

Issues began to arise when addressing how to correctly define geometric features within Knowledge Fusion to ensure that they were created appropriately. For instance, one may find multiple ways to manually create a curve or other geometric entity within the CAD system. However, for every type of geometric object to be created through Knowledge Fusion, there are mandatory parameters that must be defined to create each specific object. Therefore, the project team had to be certain that the points that are used in the construction of the gill slot were appropriately placed before they were later referenced by the Knowledge Fusion code.

Occasionally the project team was faced with difficulties in performing trim functions to the interior core splines to ensure that tangency was maintained between the existing inner core splines and the gill slot curves. At times, the students were able to appropriately perform trimming functions to the splines, and other times they were not. This variability could have been due to a loss in associativity during the gill slot creation process.

Another challenge that was confronted was the order in which the Knowledge Fusion code is processed. Typically, a command known as *demandOrder* is utilized to tell which classes should be evaluated first. However, this command was ineffective when used to instruct Knowledge Fusion on which intersection points to use in the creation of the gill slot. As a result, if NX was unable to determine where the intersection points were located, the application failed.

Last, the project team was challenged with the task of generating of a smooth curve to connect both ends of the gill slot curves on the outside of the turbine blade. Normally, this task can easily be achieved manually through the use of a command known as *Bridge Curves* within NX 2.0.

However, there is no existing functionality within NX 2.0 that allows for the creation of such a curve within Knowledge Fusion. For that reason, a normal arc was utilized to create the desired effect.

Priority 3 – Mass Properties Generator Development Overview

The Mass Properties Generator, developed in the 1970's, is an application that the aerospace manufacturer utilizes in order to analyze various characteristics of turbine blades, such as mass, volume, density, surface area, and moment of inertia. For this phase of the project, the project team was to create functionality that would allow the user to output those important characteristics of the turbine blade between two user-defined datum planes to an embedded text file within UGS NX 2.0. By doing this, the aerospace manufacturer was able to collect data about individual cross-sections and improve the balance of the turbine blades.

Mass Properties Generator Application

The Mass Properties Generator was incorporated into a standalone application as well built into the Turbine Blade Generator GUI. As a standalone application, the Mass Properties Generator allows for users to utilize the toolset without generating a turbine blade. This functionality will allow for turbine in non-native file formats, such as STEP, to be address. This application has its own variables and user parameters, making this application fully functional without the use of the Turbine Blade Generator. The Mass Properties Generator, built into the Turbine Blade Generator GUI, is for users generating new turbine blade geometry. The Mass Properties Generator, in the Turbine Blade Generator, uses other functions found in the GUI; particularly the Core Definition Tool, described in Phase 2.

Limitations found when creating the Mass Properties Generator

Limitations of the Mass Properties Generator have no bearing on not being able to address properties; it is the Knowledge Fusion code itself in limiting factor on how code can be structured. In Knowledge Fusion, a Loop() can not have more than one Class defined within it. The issues became apparent after a possible solution to create a Loop() to search and link the user defined datum plane integer to the generated and named datum planes. This defined the two boundary datum planes and then two classes were invoked to trim the turbine, creating the sectioned geometry.

Mass Properties Generator Solution

The Mass Properties Generator, both the standalone and embedded application, uses two generated datum planes as boundaries for the cross-sectional area of the turbine blade. View Figure 6 below for Mass Properties Generator overview of planes.

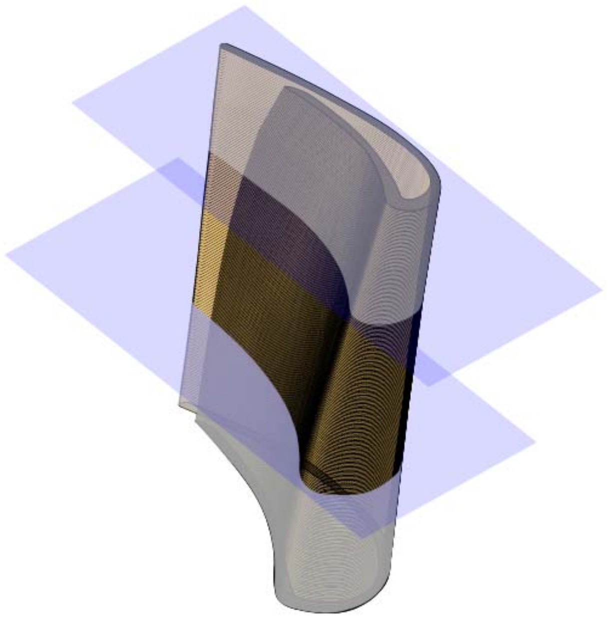
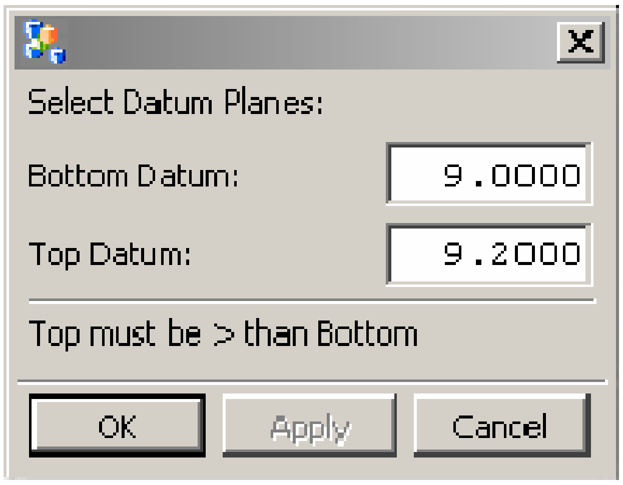


Figure 6: Illustrating the Simulated Function of Mass Properties Generator

These boundary datum planes are user-defined by the height which the user desires to analyze. (See Figure 7 below for Mass Properties Generator GUI.) The Knowledge Fusion code uses the datum planes to trim the turbine blade, creating the sectional geometry to acquire the desired properties. The output of the Mass Properties Generator follows the guidelines set forth in the 1970's format structure. The desired structure allows engineers to easily compare previous Mass Properties data sheets to those generated from the Airfoil Seed Generator. See Figure 8 for data output structure.



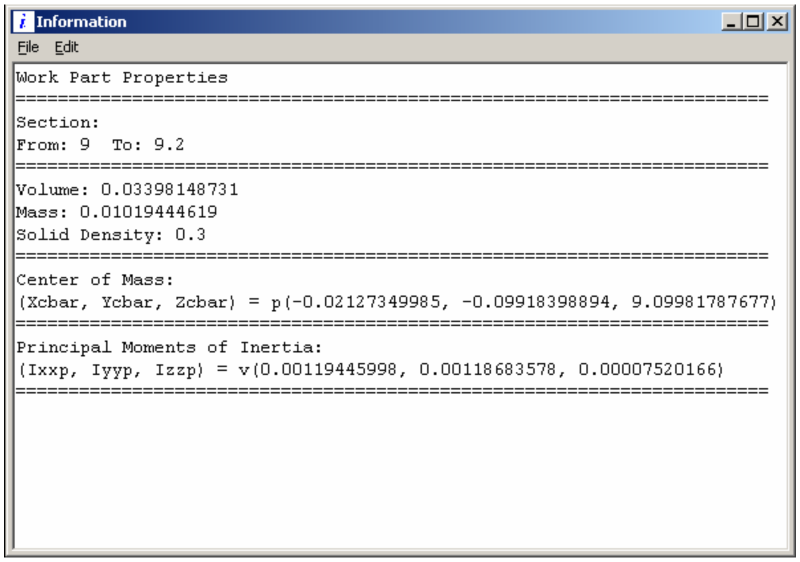
Select Datum Planes:

Bottom Datum: 9.0000

Top Datum: 9.2000

Top must be > than Bottom

Figure 7: Mass Properties Generator GUI



```
Work Part Properties
=======================================================================
Section:
From: 9   To: 9.2
=======================================================================
Volume: 0.03398148731
Mass: 0.01019444619
Solid Density: 0.3
=======================================================================
Center of Mass:
(Xcbar, Ycbar, Zcbar) = p(-0.02127349985, -0.09918398894, 9.09981787677)
=======================================================================
Principal Moments of Inertia:
(Ixxp, Iyyp, Izzp) = v(0.00119445998, 0.00118683578, 0.00007520166)
=======================================================================
```

Figure 8: Mass Properties Generator output

V. EVALUATION OF TURBINE BLADE GENERATOR

Research completed for the project enabled the team to focus on one main discipline for testing purposes to determine the overall quality of the Airfoil Seed Generator. The main emphasis was to validate the time savings of the automated geometry generated with the Turbine Blade Generator. To quantify the overall time saved when automating the creation of turbine blades, the research team worked with a Subject Matter Expert (SME) in modeling turbine blades, of the leading aerospace manufacture.

Methodology of Validation Overview

In order to confirm that the KBE solution was an effective means of automating the turbine blade CAD model, procedures were created for the SME to follow. To begin the time analysis, modeling data was provided by the Subject Matter Expert. The supplied data offered information regarding the length of time it took the SME to manually model a single turbine blade. This data concluded that an average of approximately three days, or 24 hours, was needed to complete the task of creating a full turbine blade.

The SME utilized the Turbine Blade Generator to evaluate the potential time saved. Following procedural instructions, the SME performed multiple iterations of each task using the Turbine Blade Generator. This ensured that all iterations would remain consistent and not skew the final results.

In an effort to fully analyze the Turbine Blade Generator, three separate procedural instructions were written. The areas of interest included generating a solid turbine blade, generating a turbine blade with a gill slot, and the Mass Properties Generator functionality. Information collected from each of these areas included start and finish times, indication of successful completion, and which file combinations were used.

Generating a Solid Turbine Logistics

Fifteen iterations were performed in this analysis since the majority of all turbine blade geometry generated has a hollow core. Furthermore, solid turbine geometry is typically generated first followed by the generation of the inner core using the Turbine Blade Generator. This approach would be used again during the analysis of generating a turbine blade with gill slots (See Table 1).

Solid Turbine	
Iterations:	15
Material:	15 point cloud
	15 Wall Files

Table 1: Procedure for Generating a Solid Turbine with Gill Slots Logistics

This portion of the analysis contained 30 iterations using 22 combinations of point cloud files. To complete the final iteration, 8 combinations were chosen at random. Thirty iterations were used because it allows for a good range of data collected while eliminating outliers and can be completed in a reasonable amount of time (See Table 2).

Solid Turbine with Gill Slots	
Iterations:	30
Material:	22 point cloud
	22 Wall Files
	Repeated 8 at random

Table 2: Process for Creating a Mass Properties Generator

The last time analysis conducted, the Mass Properties Generator functionality, only consisted of 12 iterations. This was due to the fact that the test was performed on a 12-section turbine blade. This meant that 11 iterations were performed by gathering the individual cross-section properties and then gathering the properties of the entire blade for the final iteration (See Table 3).

Mass Properties Generator Properties	
Iterations:	12
Material:	1 Twelve-Section Blade

Table 3: Number of Iterations for Generating Mass Properties

VI. RESULTS OF USING TURBINE BLADE GENERATOR

The results from the time saving analysis were compiled to evaluate the average mean times of the manually created turbine blade to that of the mean times of the iterations performed by the Airfoil Seed Generator. In the assessment of the project team's hypothesis, the formula H_a: $\mu > \mu_0$ was used. This formula represented that the Turbine Blade Generator (H_a) will be more significant in time savings than the manual mean time (μ_0). To test the hypothesis, a t-test was implemented. These results were assessed at $\alpha = .05$ giving the level of significance. Last, derived from the t-test, a p-value was complied giving the likelihood of the hypothesis being NULL.

Results of Generating a Solid Turbine

The data received from a SME lead to the average mean time to generate a solid turbine blade = 270 seconds (4 minutes, 30 seconds) with a standard deviation of s = 42.426 seconds. Time estimates ranged from 240 seconds (4 minutes) to 300 seconds (5 minutes). Scaled time estimates were used because the aerospace manufacture did not keep a detailed record of solid turbine geometry creation time. The SME created the basic turbine geometry and a single spline; this created the foundation and the time was multiple in respects to create a full 12 section blade.

Data measured to create the solid turbine times using the Turbine Blade Generator method measured at $\bar{x}$ = 37.93 seconds with standard deviation of s = 2.46 seconds. The total range found was from 34 seconds to 42 seconds.

It was hypothesized that the Turbine Blade Generator would produce significant time savings over the current manual method used at the aerospace facility. The hypothesis (H_a: $\mu > \mu_0$), resulted t = 7.7338. This result shows levels of significance at $\alpha = 0.05$. Derived from the t-test, the p-value at 1.303E-06; below $\alpha = 0.05$ give the chance of the hypothesis being NULL. Results displayed in Table 4.

	Current Method	Turbine Blade Generator
Mean ($\bar{x}$)	270 sec.	37.93 sec.
Std. Dev. (s)	42.426 sec.	2.46 sec.
t-value = 7.7338		
Alpha (α) = .05		
p-value = .0000013		

Table 4: Solid Turbine Analysis Statistics

This suggests that there is a significant savings in time by using the Turbine Blade Generator as to using the current manual method at the leading aerospace manufacturer. The average time savings of using the Turbine Blade Generator was found to be 85.95%

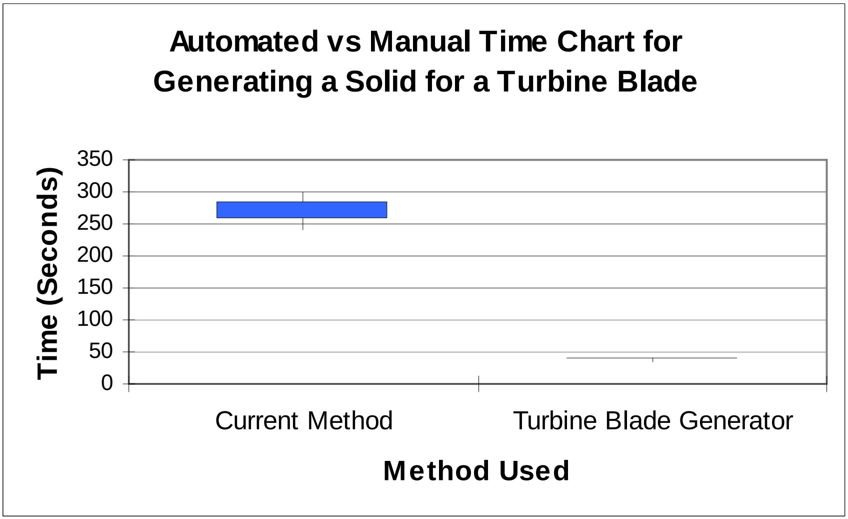


Figure 9: Box Plot of Solid Turbine

Results of Generating a Solid Turbine with Gill Slots

Data received from a SME on modeling a turbine with a gill slot was found to have average mean time to generate a solid turbine blade $\bar{x} = 21600$ seconds (360 minutes or 6 hours) with a standard deviation of s = 10182.34 seconds (169 minutes, 42 seconds or 2 hours, 49 minutes). Times ranged from 14400 seconds (240 minutes or 4 hours) to 28800 seconds (480 minutes or 8 hours).

Data measured in creating the solid turbine with a gill slot, using the Turbine Blade Generator method, measured at $\bar{x} = 60.2$ seconds with standard deviation of s = 19.6 seconds. The total range found was from 39 seconds to 125 seconds.

As hypothesized that the Turbine Blade Generator would produce significant savings again over the current manual method used at the aerospace manufacturer. The hypothesis (H_a: $\mu > \mu_0$), resulted t = 2.9916. This result shows levels of significance at $\alpha = 0.05$. The p-value resulted 0.0055; well below $\alpha = 0.05$ give the chance of the hypothesis being NULL. These results are displayed in Table 5 below.

	Current Method	Turbine Blade Generator
Mean ($\bar{x}$)	21600 sec.	60.2 sec.
Std. Dev. (s)	10182.34 sec.	19.6 sec.
t-value = 2.9916		
Alpha (α) = .05		
p-value = .0055		

Table 5: Turbine with Gill Slot Analysis Statistics

This suggests that there is a significant savings in time by using the Turbine Blade Generator as to using the current manual method that the aerospace company utilizes. .

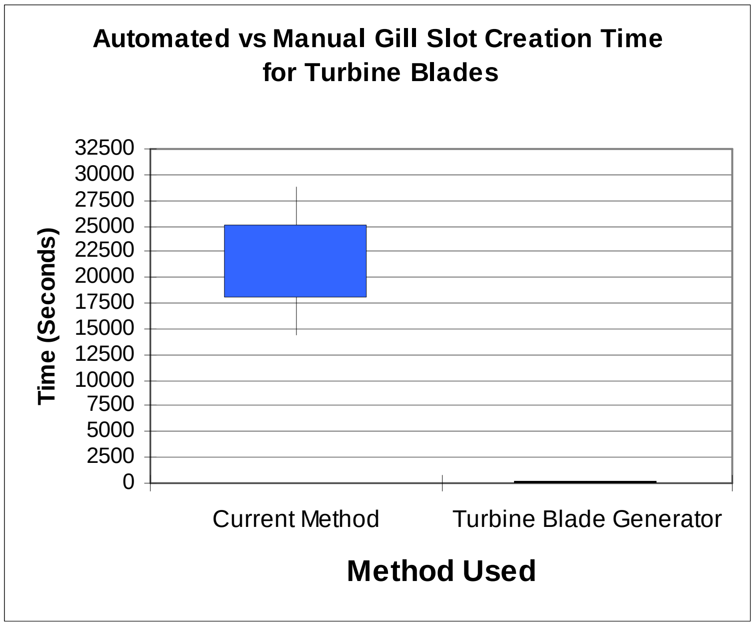


Figure 10: Box Plot of Turbine with Gill Slot

The average time savings of using the Turbine Blade Generator generating a turbine blade with a gill slot was found to be 99.72%

Results of Generating Mass Properties in the Mass Properties Generator

For the final time analysis, in addressing properties, a SME stated the average mean time to generate the turbine properties is approximately $\bar{x}$ = 1800 seconds (30 minutes) with a standard deviation of s = 509.12 seconds (8 minutes, 33 seconds). Time estimates ranged from 1440 seconds (24 minutes) to 2160 seconds (36 minutes).

Data obtained using the Turbine Blade Generator method measured at $\bar{x}$ = 39.16 seconds with standard deviation of s = 14.24 seconds. The total range found was from 28 seconds to 80 seconds.

As hypothesized that the Turbine Blade Generator would also produce significant savings in time over the current manual method used at the leading aerospace manufacturer. The hypothesis (H_a: $\mu > \mu_0$), resulted t = 4.89. This result shows levels of significance at α = 0.05. Derived from the t-test, the p-value at 0.00037; well below α = 0.05 give the chance of the hypothesis being NULL. Results displayed in Table 6.

	Current Method	Turbine Blade Generator
Mean ($\bar{x}$)	1800 sec.	39.16 sec.
Std. Dev. (s)	509.12 sec.	14.24 sec.
t-value = 4.89		
Alpha (α) = .05		
p-value = .00037		

Table 6: Solid Turbine Analysis Statistics

This shows a significant time savings by using the Turbine Blade Generator as opposed to using the current manual method at the leading aerospace manufacturer.

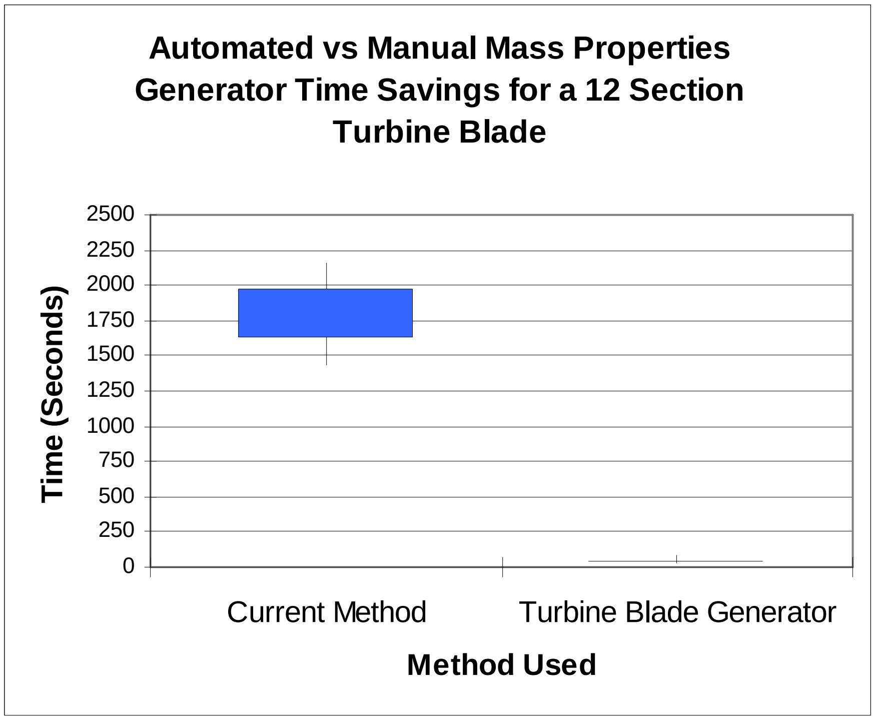


Figure 11: Box Plot of Solid Turbine

The average time savings of using the Turbine Blade Generator was found to be 97.82%

Functionality − Survey − of − Turbine − Blade Generator

At the time the Airfoil Seed Generator was developed and for the current project, Turbine Blade Generator, of spring 2007, one of the leading aerospace manufacturer employees is using Turbine Blade Generator toolset. In the beginning of the development of the Airfoil Seed Generator, the SME was able to provide input into the interface and the functionality of the tools. The Turbine Blade Generator was based of the same layout, but to quantify the usability of the Turbine Blade Generator, a survey was filled out by a SME.

VII. DISCUSSION AND FUTURE DEVELOPMENT

The results of the testing processes indicate a significant time savings for the leading aerospace manufacturer when using the Turbine Blade Generator as opposed to the current modeling method.

From the results, it is recommended that the leading aerospace manufacturer upgrade their version of UGS NX 2.0 to UGS NX 4.0. NX 4.0 has a greater amount of Knowledge Fusion functions, and thus, is superior to NX 2.0 in terms of functionality. By making the switch to NX 4.0, the occurrence of alternate solutions will be less frequent. The last recommendation to the aerospace manufacturer is to implement this tool company-wide as a standard practice for modeling turbine blades in the future.

The aerospace manufacturer has provided a list of other potential projects for the future development of the Turbine Blade Generator. The aerospace manufacturer has shown an interest in automating the generation of the cooling-hole placement, located at the base of the turbine blade stalk. The cooling holes allow air to pass through the core, preventing the turbine blade from overheating. Effectively cooling the turbine blade is a crucial design aspect for the overall quality and longevity of the product.

The aerospace manufacturer also desires the additional functionality of generating a squealer tip into the Turbine Blade Generator. A squealer tip is used to alter the heat coefficient of the turbine blade. However, this feature is not created in every turbine blade model. The development of including a squealer tip option within the user interface is in the near future.

The last item that the aerospace manufacturer discussed for the future development of the application was the incorporation of a cost calculation tool. This tool will need to have the ability to be updated continuously due to the ever changing market for materials.

VIII. REFERENCES

[1] Brain, Marshall. *How gas turbine engines work*. 2007. How Stuff Works. < http://www.howstuffworks.com/turbine.htm>. Retrieved on March 21, 2007.

[2] O'Brien, William, Chandonais, Michael, Henderson, Robert, Prezbyndowski, Mark, Hartman, Nathan, & Rasche, Joseph. (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.*

[3] Golkar, Mohammed. "Development of KBE Support for Design and Analysis of Car Components Using UGS NX-Knowledge Fusion." *Lulea University of Technology*. 2006.

[4] UGS Corp. *Knowledge Fusion for Designers – Student Guide*. United States of America, 2006.

[5] Yip, Daniel C., Law, M., Cheng, K., Lau, K., Barnes, Stuart. "Knowledge Intensive CAD in Product Design Validation." *International Journal of Knowledge-Based Intelligent Engineering Systems*. 9.2. (2005): 45-61.

[6] Day, John. *Knowledge-Base Engineering – Automating for Profitability*. October 24, 2005. Design News. <http://www.designnews.com/article/ CA6275341.html>. Retrieved on February 14, 2007.

[7] Bermell, P. and 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.

[8] Prasad, Brian. (2005). What Distinguishes KBE from Automation. *COE NewsNet*. Retrieved April 27, 2006 from http://www.coe.org/newsnet/Jun05/knowledge.cfm#1.

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

[10] UGS Corporation. (2004). *Knowledge fusion for users: 2004 PLM World*. Retrieved on February 15, 2005 from pdf file.

[11] 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.

[12] 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 February 9, 2006, from ACM Portal database.

[13] 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.

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

[15] Susca, L.[1], Mandorli, L.[2], Rizzi, C[1]. (2000). *How to represent intelligent components in a product model.* Department of Industrial Engineering, University of Parma[1], Italy. Department of Mechanical Engineering, University of Ancona[2], Italy.

[16] 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.

[17] 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, February 16, 2006.

[18] Barnum, C.M. (2002). In Usability Testing and Research (pp.31-50). New York, New York, USA: Longman.