Proceedings of Global Power and Propulsion Society
ISSN-Nr: 2504-4400
GPPS Chania24
September 4-6, 2024
www.gpps.global

GPPS-TC-2024-132

GRAPH-DRIVEN INTERACTIVE PROCESSES IN TURBO-COMPONENT PRELIMINARY DESIGN

Jens Schmeink
German Aerospace Center (DLR)
jens.schmeink@dlr.de
Cologne, D-51147, Germany

Marius Bröcker
German Aerospace Center (DLR)
marius.broecker@dlr.de
Cologne, D-51147, Germany

Stanislaus Reitenbach
German Aerospace Center (DLR)
stanislaus.reitenbach@dlr.de
Cologne, D-51147, Germany

Martin Siggel
German Aerospace Center (DLR)
martin.siggel@dlr.de
Cologne, D-51147, Germany

ABSTRACT

The objective of this work is to introduce an interactive design process for turbo machinery components, utilizing a graph-based framework. This approach integrates various methodologies, typically involved in the preliminary design phase, into a cohesive structure. The process steps are represented as nodes, while the flow of data is depicted through connecting edges. A key feature of this method is the inclusion of both pre- and post-processing steps within the graph. This integration not only facilitates a more holistic view of the design process, but also enhances the designer's ability to trace cause-and-effect relationships within the workflow. The interactive environment thus created allows for rapid evaluation and effective decision-making. The entire process is designed to be non-monolithic, offering significant advantages in terms of flexibility and adaptability. Further, the modular structure allows for seamless integration into broader systems, such as optimization routines, data analysis procedures, or the generation of surrogate models.

INTRODUCTION

The design process of a product is characterised by a digital representation of the future result. This virtual product is manipulated, improved and tested using various tools until the design is finalized. Some applications are characterized by very large, complex processes with a very high degree of automatization. However, there are processes carried out through manual operations and benefit from a high degree of interactivity in the design software. The procedures presented in this paper are intended to address the latter. However, the possibility of full automation is always provided.

The graphical representation of various steps in the design process has become an integral part of an engineer's everyday work. The range of processes is very broad: both in terms of depth of detail and disciplines, but also in terms of execution and utilization. An application study about the effect on specific design principles for user interfaces for an engineering design interface was done by Rothrock et al. (2006). Following the *proximity compatibility* (PCP) and the *control-display compatibility principles* (CDCP) (Wickens and Carswell, 1995) they showed positive effects of proximity of design control elements among each other, but also the close connection of an control and its visualization. Simpson et al. (2007) explored the effects of delay time and different types of *richness* of a GUI on the design efficiency of users. Their description of the *richness* is based on how information is processed and what additional indirect information results from this. It was shown that an increasing *richness* has an positive influence of the decision finding process of an engineer. As an example, it is shown that two numerical values in simple text fields do not visually convey any further information. In comparison, the representation of the same values in a bar chart can also convey the relationship between the values. As a result, the bar chart provides a higher *richness*.

The use case for the presented approach is the preliminary design of an axial compressor. There is a broad market segment for this application. Due to the proprietary nature of many of these tools, insights into the details of their functionality are not available. Notable examples with a graphical user interface are TURBOdesign1 from Advanced Design Technology (Zangeneh, 2010) AxSTREAM®(Romanenko et al., 2008) from SoftInWay, AXIAL® and AxCent® from Concepts NREC (2019) or the program elements of PCA Engineers (Vista ™Toolbox) for through flow analysis(Casey and

Robinson, 2010). These programs are specialized in their field of activity and therefore offer a great depth of models and breadth of presentation of results. In addition, in several industrial applications, models (e.g. by Wright et al. (1991)) are used in established processes without or with a graphical user interface.

The approach shown here is intended to map the functionalities of the compressor preliminary design by means of graph-based processing and thus make it flexible and interactive.

Graph-based modeling has proven itself in numerous simulation programs - in particular through the Modelica model language (Mattsson and Elmqvist, 1997) and its usage in software such as Modelon Impact (Modelon, 2020) and Dymola (Dempsey, 2006). In these tools, the (physical) model is represented by means of a graph. The approach of process modelling via nodes as part of graphs is currently a trend that is being adopted for various use cases and is also pursued in this work. The software Blender (Blender Development Team, 2022) for generation of 3D visualizations is a well known example using a system of connected nodes to define e.g. shading or composition. Knime (Berthold et al., 2009) is a software program that uses this approach for data analysis, while Grasshopper®(McNeel et al., 2014) and Synera (formerly known as ELISE) (Maier, 2011) focuses on computer-aided design /engineering (CAD/CAE) interfaces. The ANSYS Workbench(Chimakurthi et al., 2018) also employs an approach where processes are linked to graphs. The focus there is on the processes of data transfer of geometries and tools of structural mechanics or CFD analysis.

Moreover, the graph-based design process has already been used in the same software framework used for this study: Siggel et al. (2024) utilized the graph-based approach for the post-processing of a data model-driven geometry in preparation for further processing for high-fidelity computational fluid dynamics (CFD). Reitenbach et al. (2024) investigated the automated generation and modification of node-based workflows using a large language model (LLM). This paper deals with an application of the approach for the design of a compressor as part of an aircraft engine with focus on preliminary design, but also addresses the links to high fidelity processes.

A development engineer usually has a toolbox of established process modules for various design or simulation steps. The connection of data and process steps with each other is heavily dependent on the tool landscape of the company or organization. This paper presents a system that uses the low-code approach (Richardson et al., 2014) to flexibly design and link processes together in form of graphs without restricting experts in their options by supporting scripting interfaces.

The focus of presented approach is the interactivity while maintaining the option of the usage of the designed process in broader systems, such as optimization routines or parameter studies. Its purpose is to enable the engineer to develop processes that make it possible to link cause (design decision) and effect (impact on relevant characteristic values and processes) as effectively as possible in order to make well-founded decisions quickly. A series of graphical reports are made available in the system via nodes in order to integrate them directly into the design decisions. These aspects also makes the process appear advantageous for utilization in the induction of new employees or in education. Process design and graphical representation work seamlessly together in this approach, enabling a user interface design that fits the problem and positively influences the engineers according to the design principles mentioned above. The findings will be used to develop best practice recommendations for the architecture of the process in an interactive preliminary design system.

The use case shown here is based on the fact that the individual process steps can be processed in a short time (single-digit number of seconds). In principle, the system is also suitable for linking more time consuming processes, although the graphical representation of the results following the processes naturally does not provide such rapid feedback on the user input. For such a system, the evaluation can be accelerated by means of surrogate models. These can also be integrated into the system, but will not be explored further in this paper.

METHODOLOGY

This section describes the principles of the underlying graph system. In addition, some of the nodes that play an essential role in the interactive preliminary design processes are presented.

Graph Based Process System

The interactive design environment is implemented in the open-source software framework GTlab (Reitenbach et al., 2020, 2023). GTlab provides a software platform for the collaborative and multidisciplinary design and simulation of propulsion systems. The modular architecture allows the framework to be extended with additional functionalities in form of modules. These modules can implement and integrate various functionalities, such as discipline-specific models and simulation procedures, process elements, and tools. Further, fundamental changes to the framework, such as the addition of scripting capabilities, are realizable using modules.

Based on the QtNodes library (Pinaev, 2017), the possibility of building directed acyclic graphs was integrated into a module to combine individual processing procedures into unified workflows. In this system, called IntelliGraph, these procedures are represented by the nodes of a graph. A processing procedure could be as simple as basic mathematical operations, as shown in 1, or as complex as fully functional fluid simulations. The edges denote the connections between nodes and, thus, the flow of data and information. For this purpose, each node has at least one input or output port representing the input arguments of a node and the result data respectively. To interconnect two nodes, one output port must be connected to another node's input port. In general, each output port may be connected to any number of input ports of other nodes

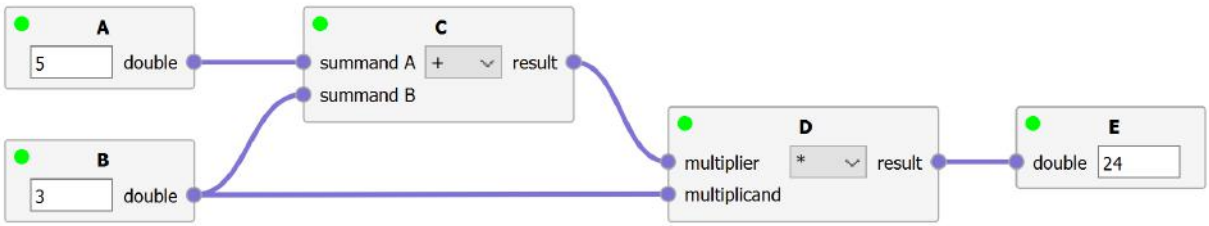

Figure 1 A simple graph performing exemplary mathematical operations

On the logic level, the data flow is implemented in the following way: Each port is associated with a data object, that holds some data or information. A data type defines the type of data a port can hold. Only ports that have identical or compatible data types can be connected. Data types are graphically reflected by uniquely colored connections and ports, allowing the user to quickly distinguish incompatibilities. Regarding data management, all connections that share the same output port of a node should have access to the same data by definition since, generally, data is produced at output ports and consumed at input ports. For this purpose, a shared ownership relation is employed in which a single data object at an output port may be accessed by all of its connections and, by extension, by all input ports of the connected nodes. Since a data object is potentially shared between multiple connections, the data object is made immutable once created, allowing for read-only access but prohibiting modifications to the data. If a processing procedure of a node has to modify the input data, e.g. the node outputs a modified version of the same data type, a deep copy has to be made within the processing procedure. Thus, copies are only made when required, allowing for a more efficient data exchange between nodes.

The data flow is only possible and defined in one direction between the nodes. Furthermore, loops in the information flow are not supported to not introduce additional complexity into workflows. Still, iterative behavior is supported by either evaluating an entire graph multiple times with different inputs or using a single node that evaluates its own processing procedure iteratively.

When evaluating a node or the entire graph, data dependencies must be considered. The dependencies of a node can easily be deduced by following the data flow upstream: all nodes that are connected to the input side are considered direct predecessors or the dependencies of a node. To keep track of whether a node's dependencies have been satisfied, the graph stores the evaluation state of each node and its ports. In the beginning, all nodes have not yet been evaluated, so the data on all ports is outdated/invalid. To start the evaluation, all nodes that have no predecessors are evaluated. Once a node completes its evaluation successfully, its output data is considered valid. All its successor nodes are then triggered, but are not evaluated until all of their input data is valid, i.e., all dependencies are satisfied.

Accordingly, each node is evaluated exactly once. If a node changes, e.g. because its input(s) changes, the node itself and all of its successor nodes are invalidated and reevaluated as described before. Redundant and potentially costly evaluations can be avoided, which is crucial for a graphical tool that leans into responsiveness and interactivity. If a node fails its evaluation, its successor nodes are not triggered, and the output data is considered invalid. Furthermore, the node is indicated as failed graphically. The node is only reevaluated once a predecessor changes or if the user explicitly triggered the node evaluation once more. An alternative to automatic node execution is manual execution mode. In this mode, the principles remain the same, but each node must be executed manually. Additionally, nodes can be paused to prevent execution in automatic mode. Both features are particularly useful during graph development or when handling nodes with long run times.

Since data dependencies are clearly expressed and processing procedures are encapsulated within a node, a graph-like processing workflow can be parallelized. Nodes, that have no dependencies between each other, can be executed concurrently. This can reduce the total execution time of more complex graphs significantly. In addition, the parallelization allows nodes and their connections to be modified or deleted during execution without causing unwanted side effects. If a node was altered during evaluation, the result is discarded, and the node is re-evaluated. However, some nodes may not be evaluated in parallel since they may access or modify the same or a restricted resource. To handle such a case, a node can specify on the programming level that it should be evaluated exclusively to other nodes. The graph system respects this information when evaluating other nodes without additional synchronization barriers.

The implementation of a node is very flexible and can be customised as required. Input- and output ports are defined at compile time but may also be altered, added, or removed at runtime. If desired, the graphical representation of a node may be changed to better suit a specific application. Furthermore, nodes may also register graphical elements to change the visual representation of the node. Such widgets may be applied to display the internal configuration, visualize the output data, or allow the user to modify how data should be processed within the node. This enables a developer to implement powerful and specialized nodes.

IntelliGraph is implemented so that a graph itself may be part of another graph, effectively creating group nodes. In

this case, a so-called input- and output provider is added to the sub-graph, denoting the group node's input- and output data. Using sub-graphs, complex workflows may be grouped into a single node, abstracting implementation details and reducing the complexity in the top-level graph. Furthermore, the logic of a graph may be reused in other graphs, allowing the creation of templated workflows. From the user's perspective, a sub-graph behaves exactly like any other node, except that the user can investigate and modify the configuration.

Graphs, as described in this publication, are built using the object system of GTlab. Based on this foundation, each node can define and register certain properties. These may be used to incorporate specific settings and configurations that should be saved persistently.

In this publication, the system is applied to the preliminary design of a compressor. For this purpose, a series of specific nodes were implemented to map individual process steps. The system also offers the possibility of embedding graphical elements in the nodes. In this way, the results of intermediate steps can be easily visualized to provide feedback beyond numerical values.

Elements of the process workflow for component preliminary design

Based on the described functionality of *IntelliGraph*, the following section presents nodes that are of particular importance in the interactive preliminary component design. Several aspects of node design have proven to be generally useful. Similar to the UNIX philosophy (Raymond, 2003), a clear recommendation has emerged that nodes should be designed to focus on one task only in the process, e.g. manipulating a specific set of data or running a simulation. This has several advantages. First, it tends to reduce the runtime of individual nodes, which improves interactivity. Second, such nodes are much easier to reuse in other processes. Another suggested best practice is to use subgraphs when nodes are assembled multiple times in a typical combination. These can then be easily reused without the need to create monolithic blocks for such tasks; instead, it is possible to continue working with the flexible small-part nodes.

Geometry Model Generation

To ensure consistent modelling and handling of data, the preliminary design process considered employs a common data model. This data modelling approach allows a turbo machinery component to be uniformly described by a 2D representation of the strut geometry, including the positioning of the blades (Reitenbach et al., 2021). This can be extended with tool-specific information, e.g. profile description for the aerodynamic evaluation. By using a standardized interface to access model information, file I/O can be avoided, allowing model data to be exchanged between connected tools in a standardized manner. Data model components that are to be used in the process chain can be copied from the existing data model via referencing or generated within the process chain. Of particular importance in this work is the generation of a 2D compressor model based on the approach of Häßy and Schmeink (2022): The principle of this approach is the parameterisation of the geometry based on dimensionless characteristics. This includes spline definitions for mean line and height profiles, and a series of tables for the 2D shape of the blades. These can be determined, for example, from the analysis of an existing 2D geometry. For the generation of a new geometry model based on this characteristics additional information are introduced into the process as the input and output area of the new component.

Additionally, an alternative specification of the input parameters is feasible, utilizing splines for the tip and hub contours rather than the mean line and height profile. Similar approaches have also been successfully used for compressor pre-design by e.g. Keskin (2006) and Hinz (2012).

The node that introduces the corresponding geometry generation process into the graph system has input ports for each of the spline and table objects to be used and an output port for the resulting data record representing a turbo component. The additional parameters are introduced as properties of the node which make them accessible for the user in the interface but not in the graph via ports, as this would compromise the graph's clarity.

Another node that is important for the interactive design is the Python node. This allows flexible intervention in the process via scripting. Parameters can be integrated into the Python environment of the node via freely definable ports which can be further used in the graph. This allows prototypical or very application-specific manipulations in particular to be integrated into the graph. Despite the freedom of scripting, scripted nodes should also focus on one task only to improve traceability and reusability.

The ability of the nodes to use a custom visual representation is of great importance for the inclusion of pre- and post-processing in the graph and thus the design process. For the manipulation of spline objects and tabular relations, nodes are implemented for visualization and manipulation via control points (cf. Fig. 2) The manipulation can be done in the node using drag'n drop or by entering specific values.

Simulation

The aerodynamic evaluation of the compressor design is carried out by means of the 2D through flow code ACDC (Schnoes and Nicke, 2017). A particular node running a full through flow evaluation has a port for entering a compressor data record, a port for transferring the thermodynamic boundary conditions and a port for an optional directory path to store additional result files. A graph using this node is shown in Figure 7. The thermodynamic boundary conditions can

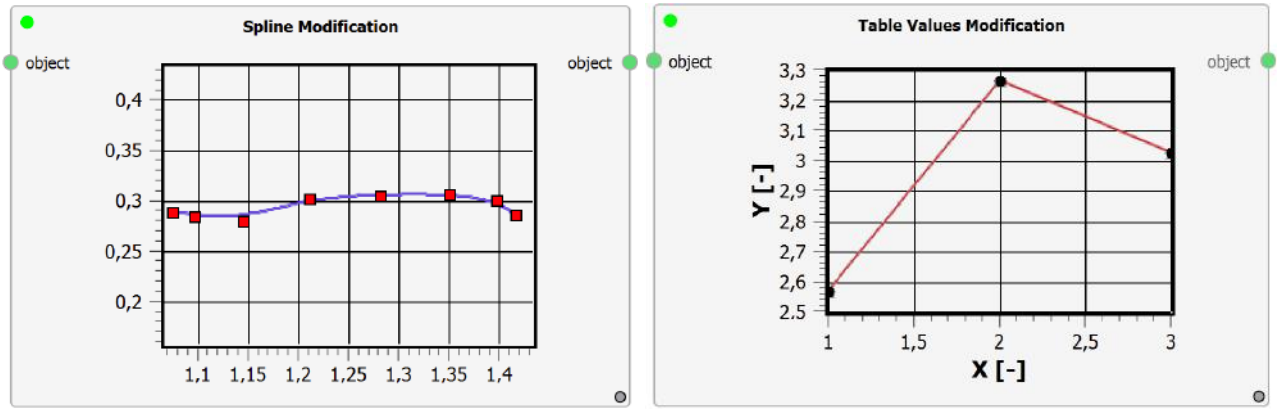

Figure 2 Nodes for spline and table values visualization and manipulation

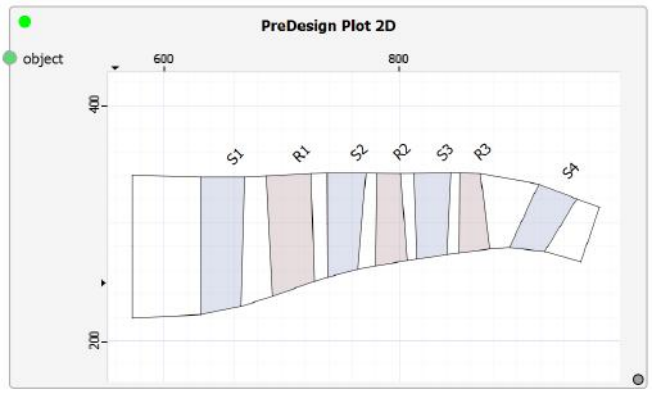

Figure 3 Node for the visualization of 2D data model representation of the given component

be defined in a separate node, which combines the numerical values into a common data set and communicates with the ACDC node via an edge. This approach has proven to be a good practice. On the one hand, this improves the clarity of the simulation settings, as modelling parameters and boundary conditions are no longer set in one window. On the other hand, this makes it much easier to reuse such an operating point definition. For example, if a number of design options are to be evaluated in several ACDC nodes, it is easy to ensure that identical values are used if they are all connected to the same input for the boundary conditions. A text field on the node informs the user whether the simulation has converged, not converged or has invalid input parameters. After the simulation a result table is attached to the previous selected compressor object and this supplemented data record is offered for further analysis via an output port. Further output ports allow to access characteristic result variables, such as efficiency or pressure ratio for a fast assessment or visualization.

Evaluation

A special feature of the presented approach is the inclusion of the visualization of results in the process chain. For this purpose, various nodes were implemented in the system. The balance between general validity of the nodes and specification to improve the information value has to be evaluated for each new implementation. The routines in the nodes are designed to be reusable for different applications. However, for the evaluation of very specific results or result structures, for which reports have already been established, these have been transferred to the graph-based process design system. To enable a rapid and straightforward evaluation of a 2D geometry model, the system provides a node to plot 2D geometries, as depicted in Figure 3.

Nodes were also implemented for an equally suitable evaluation of the aerodynamic simulation. A node representation of the characteristic values over the radial height of the blade rows was created to provide detailed feedback to the designer, such as the influence of loss models (cf. Fig. 4).

A dedicated contour plot node is provided to visualize the distribution of various flow parameters in the radial and axial dimensions as a result of a simulation (cf. Fig. 5)

APPLICATIONS

The following examples demonstrate the utilization of interactive processes in compressor design and analysis. They include basic graph applications for sketching and simulation, interactive result processing for improved design decisions,

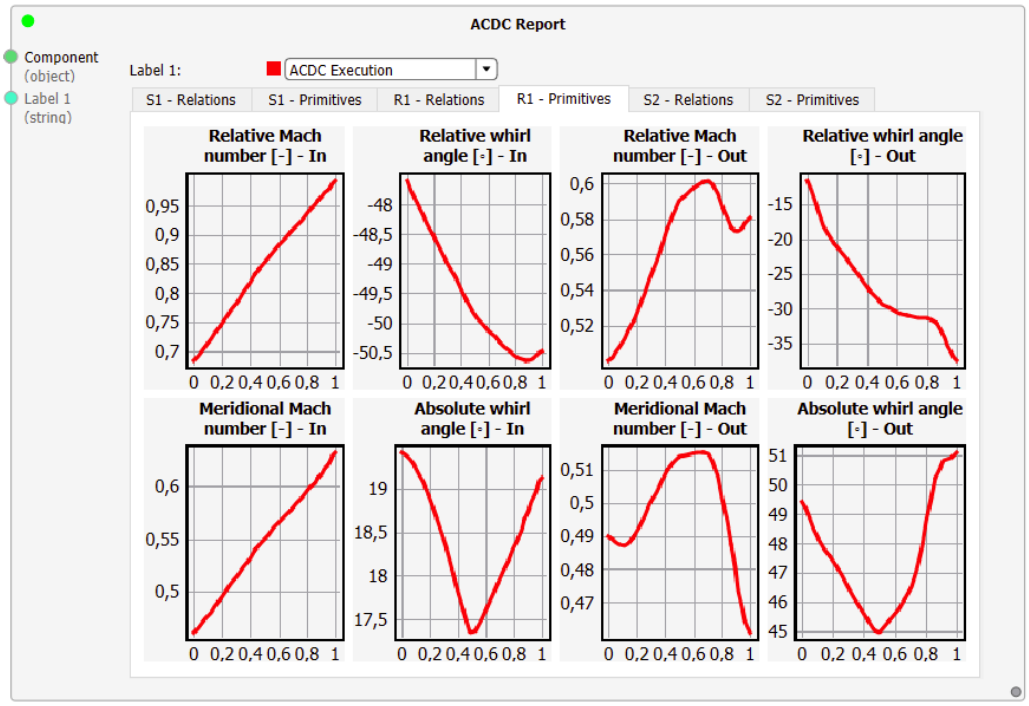

Figure 4 Node for the visualization of aerodynamic results of the blade rows in radial direction

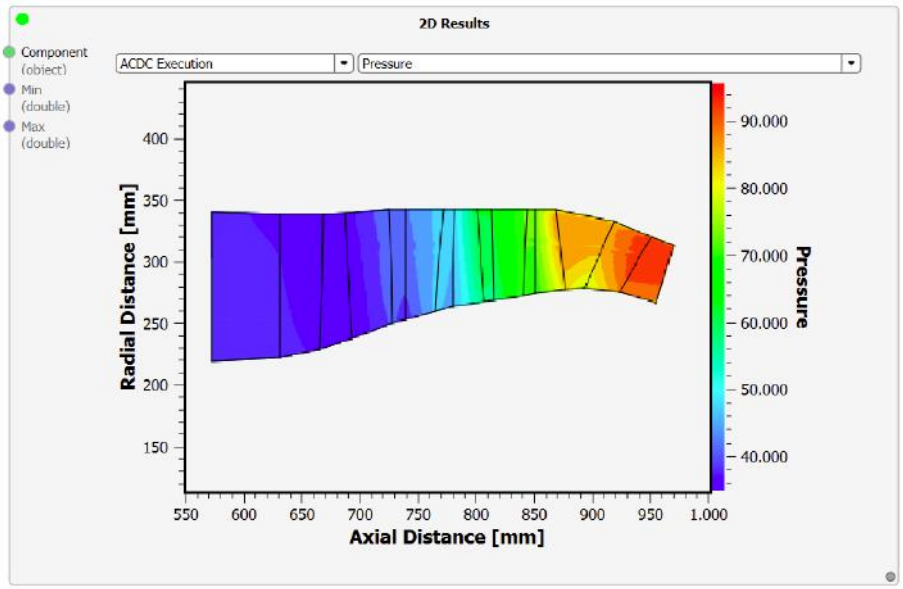

Figure 5 Node for the visualization of aerodynamic results in two dimensions

and processes that go beyond the preliminary design. The option of combining nodes as sub-graphs is applied in all use cases. In the figures, these nodes can be identified by the dark grey colouring.

Basic Applications

This section presents two basic process chains that are fundamental for more complex configurations and provide a first impression of how they can be applied on the interactive design process:

Initially, the geometry generation of the 2D representation of the compressor is highlighted. The process is mapped as a graph and is shown in Figure 6. The graph consists of nodes that can be divided into four stages. First, existing spline data sets are referenced via two nodes. These splines are visualized in the second stage and can be manipulated using the control points. A geometry generation node is called up in the following sub-graph. The resulting compressor data record is finally visualized in a two-dimensional representation in the last node.

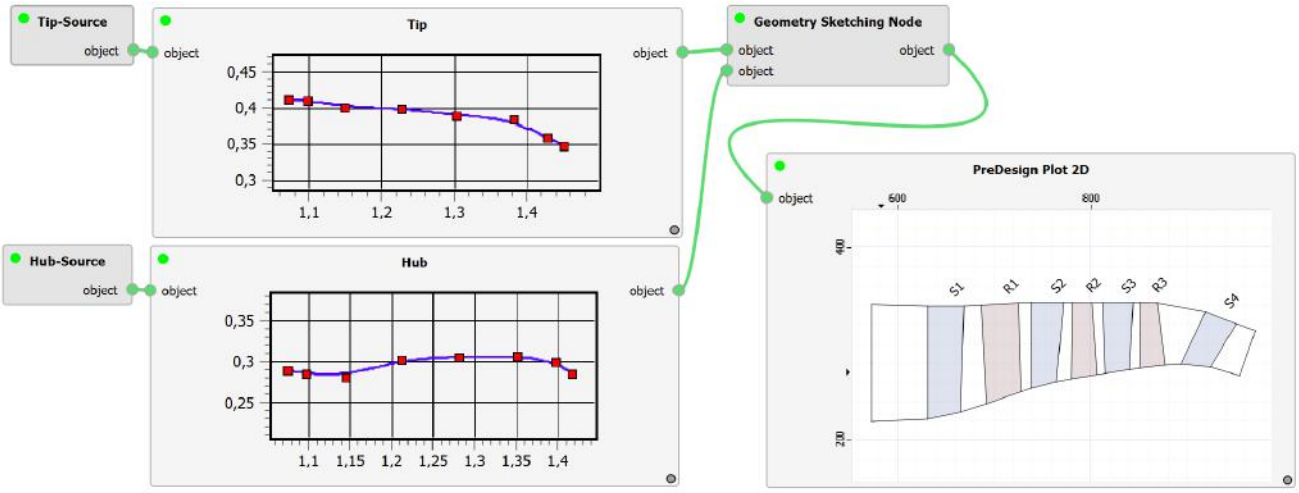

Figure 6 Geometry generation graph

The interactivity of the process here lies in the possibility of manipulating the splines to describe the tip and hub contour and the resulting automatic evaluation of the nodes downstream: A manipulation of the nodes thus directly triggers a new evaluation of the geometry generation and its visualization. Important advantages of the used approach by Häßy and Schmeink (2022) are the short run time (less than one second) and its robustness, allowing a wide range of changes to be processed. In this way, the user can gain an immediate impression of how changes affect the geometry model. The control and visualization elements follow the aforementioned principles of GUI design.

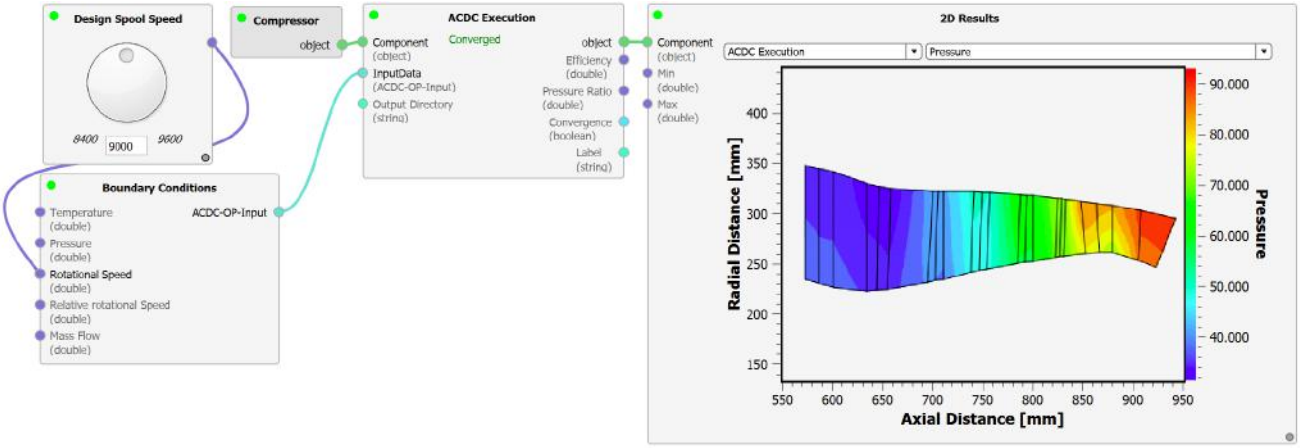

Figure 7 Aerodynamic simulation with ACDC organized in a graph

Figure 7 shows the aerodynamic analysis application case using the tool ACDC. The graph starts with the specification of inputs, followed by the aerodynamic analysis and finally the graphical representation of the results. A node is used to enter the thermodynamic boundary conditions of the simulation, for which the values can be set via properties in a separate window in the framework or alternatively via a port. In the presented example, the design spool speed is defined through a control element node, while the remaining parameters are set via properties. By defining suitable limits for the control element by the process engineer, the correct usage by others can be ensured. Input variables are transferred to the node for the ACDC execution together with a data representation of a compressor. After a successful simulation, the compressor object is passed to the visualization node along with the result data. In this example, the user is able to quickly identify any changes in the results when the design speed of the compressor design is changed. Similar investigations could easily be added for other parameters, e.g. the mass flow.

Interactive Process Design

The focus of the previous sections has been on the basic process of using graphs. How this can influence the preliminary component design through interactivity has only been discussed in principle. This will be illustrated below using an example based on the processes presented above.

The graph for this process is illustrated in Figure 8. There are three parts: geometry generation, analysis with ACDC,

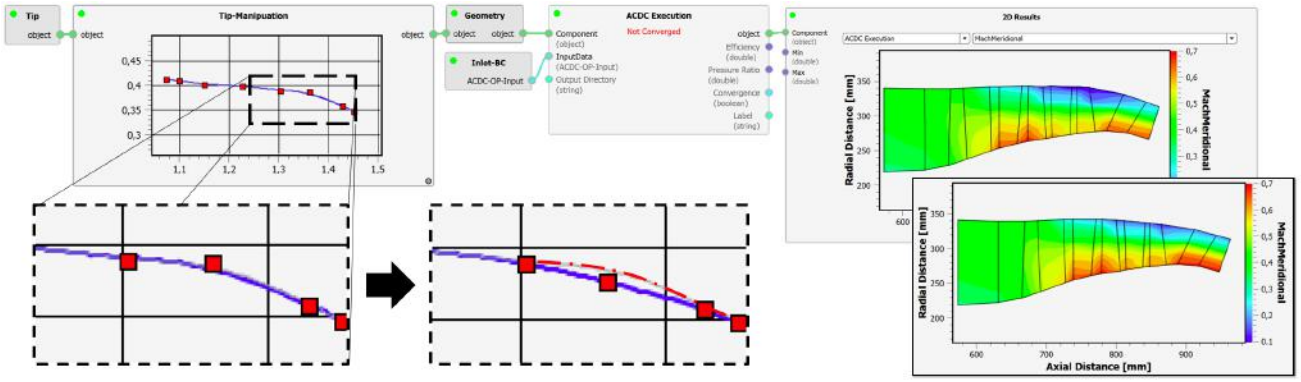

Figure 8 Interactive design with manipulation of tip line to improve convergence of the aerodynamic simulation. Red dashed lines shows the original curve compared to the new blue manipulated spline curve

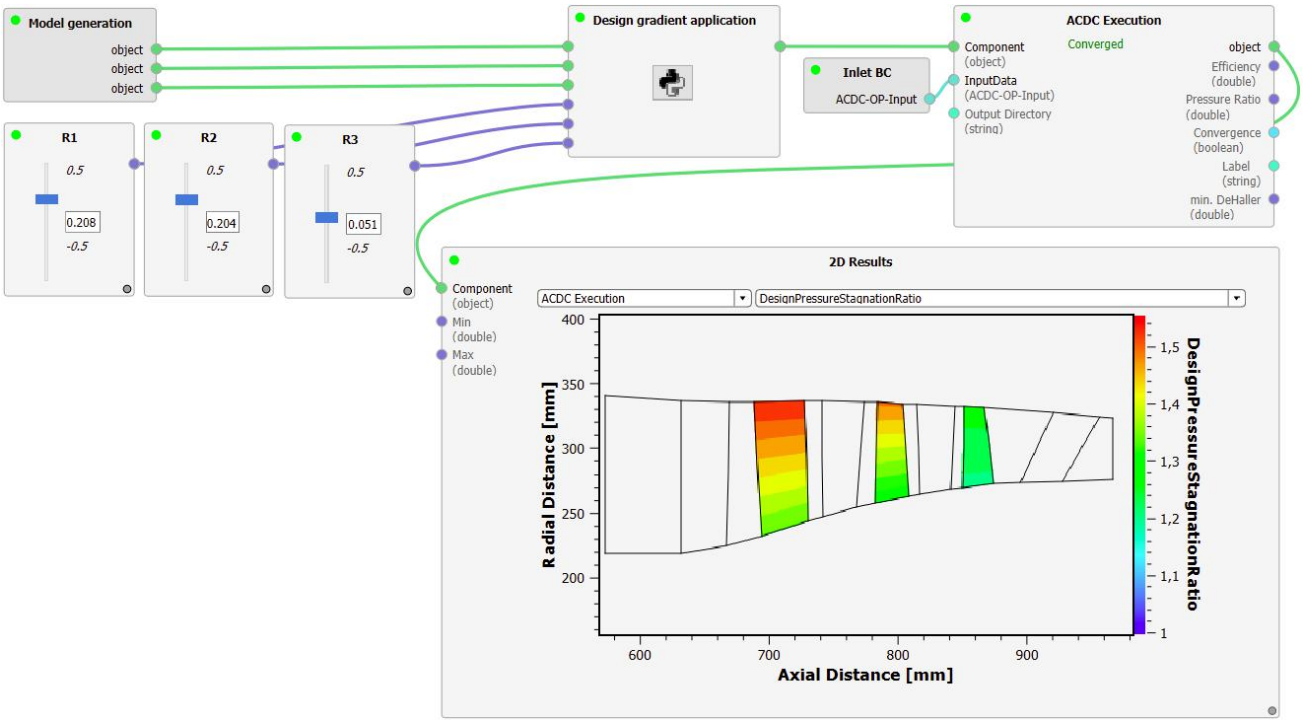

Figure 9 Example for a graph with explicit increased richness

and visualization of the results as a contour plot of the meridional Mach number. The geometry generation process is simplified using sub-graphs so that only the tip contour manipulation node is shown in the main graph. The sub-graph includes additional inputs that are held constant.

The analysis of the initial compressor model indicates that the solver does not converge to a solution for the defined operating point. This is where an interactive design provides the advantage of allowing the engineer to intervene in a targeted manner. The result visualization node provides an insight into the first (non-converged) flow solution. Low values of the meridional Mach number at the tip in the rear third of the compressor model are an indication of a possible cause of the poor convergence behavior. Expert knowledge can now be quickly and purposefully incorporated into the process. The contour of the tip shape can be intentionally adjusted to address the low meridional Mach number region. Changing these contours triggers a new analysis with ACDC and updates the results. A small manipulation of the tip contour is sufficient to obtain a convergent solution in this case.

The following example demonstrates how graphs and their visualizations can enhance the GUI's richness, enabling engineers to make quick line-oriented decisions. A corresponding graph is shown in Figure 9. This process can also be divided into model generation, simulation and evaluation. In this case, one design parameter of the rotor profile description for ACDC is radially distributed separately using a mean value $\Pi_{mean,i}$ and gradients Δ_Π for each blade row i. A decreasing mean value is applied along the compressor from one rotor row to the next. The python node applies these gradients and mean values to determine the values for all relative radii based on a linear distribution from hub to tip.

The gradients are defined in nodes and then transferred to the Python scripting node, which modifies the compressor data record. Gradient values are not only controlled through direct numerical input, but also through sliders. This graphical representation (in accordance with the GUI design principles presented above) helps to achieve a user interface of a higher *richness*. The user not only sees a numerical value, but perceives it in relation to the reasonable limits suggested by the expert. This effect is reinforced by the contour plot at the bottom right of the illustration. The design parameter is presented in the plot, which shows the color gradients for each blade row and the resulting parameter values along the compressor mean line from one rotor blade row to the next. With this impression, modifications to the design parameters can be carried out in a more informed manner in order to improve the flow solution. As a best practice, we recommend that only those values that are actually intended to be changed are offered for manipulation by prominent nodes.

High-Fidelity Processes and the Link to Detail Design

Preliminary component design is one of the first steps in the overall component design process. This is followed by high-fidelity (Hi-Fi) processes such as 3D CFD or structural analysis. Compared to the lower-fidelity algorithms, the setup of Hi-Fi simulations typically requires more user interaction and therefore benefits from an interactive workflow system. In particular, the integration of these simulations requires a geometric model of the engine components to be analyzed. These Hi-Fi processes typically include a mesh generation step followed by the actual simulation code. In a first step, the corresponding geometry information must to be extracted from the preliminary design results using shape generation algorithms implemented for the different turbo-component parts. In a second step, these geometries often have to be post-processed to meet the requirements of the mesh generation software, e.g. by labeling certain surfaces or creating auxiliary geometries. A graph-based system like *IntelliGraph* is ideally suited to define a post-processing geometry pipeline. First, arbitrary data can flow between the different nodes, including CAD geometries. Second, it offers flexibility by providing a versatile set of tools to modify and customize the geometry modification. And third, it provides process automation, since these processes need to be repeated for design changes. An example of the geometry generation process is shown in Figure 10, where the thickness-to-chord ratio of the rotor blades is interactively modified and which has a direct impact on the resulting blade shapes. In addition to changing the overall blade thickness, the graph contains other steps such as adding a fillet parameterization to the original compressor blade parameters and a geometry generation step using GTlab's geometry module. Note, that some process nodes are grouped together to focus on the main aspect of the interactive relationships between a design parameter and the resulting 3D geometry. Due to the visual representation of geometry directly in the graph, the effect of design parameter changes on the 3D geometry is eminent.

Many turbomachinery preliminary design programs offer the option of transferring designs from preliminary design processes to 3D CFD simulations. One such option is available for the process presented here for the transfer to the Auto-Grid™ mesher for turbo machinery applications and thus to downstream CFD simulations. A special feature, however, is that the 3D representation of the results can also be modified in the same system, enabling much more specific work with the CAD models, as explained subsequently. Figure 11 shows a custom post-processing to prepare geometry for a Hi-Fi CFD analysis. First, a 3D geometry of a turbine blade is created, which is then used in a second step to create the fluid volume through the annulus around the blade. Subsequent steps — not shown here — include exporting the geometry to a CAD exchange format, a mesh generation step based on the exported file, and the final CFD analysis. Although detail design is not currently within the scope of the system presented, the link to detail design is considered. Geometric metadata can be set and modified in the node system and the final geometries can be exported using specialized CAD exchange nodes. Since geometric metadata is preserved even after geometry modification and CAD exchange, shapes and sub-shapes can be uniquely identified in a following parametric detail design, enabling a tight coupling of preliminary and detail design. More details on the metadata system and the integration with detail design can be obtained from Siggel et al. (2024).

DISCUSSION AND CONCLUSIONS

A process system was demonstrated that can be used to build a design process chain by a flexible combination of process elements. Visual feedback is provided to the user by displaying results directly in connection with the input parameters or process steps. The flexible composition of the nodes, including the visualization of the control variables and the results, is used to create a suitable user interface for the respective application and to comply with corresponding design principles. In this way, the engineer is offered a valuable opportunity to interactively influence the design and to recognize the relationships between cause and effect. This possibility of gaining an insight into the system is of particular interest for manual design, for training new employees and for teaching. In addition, some recommendations were developed on how to design nodes and graphs in such a system for the interactive preliminary design. Another advantage is that the system is very well suited to defining standard processes in the form of graphs. Experts knowledge can be incorporated into the process design and evaluation and further utilized. The nodes are defined in a way that they do not affect the flow upstream. This allows for easy individual extension of the standard, such as through additional visual evaluations, while maintaining its basic structure.

Even though the presented approach is generic compared to specialized software, several features of specific programs are also transferred to this application. Examples include a procedure comparable to the "inverse design" of TURBOdesign1 (Zangeneh, 2010), in which flow variables are usable as design parameters, or the ability of many programs to transfer the

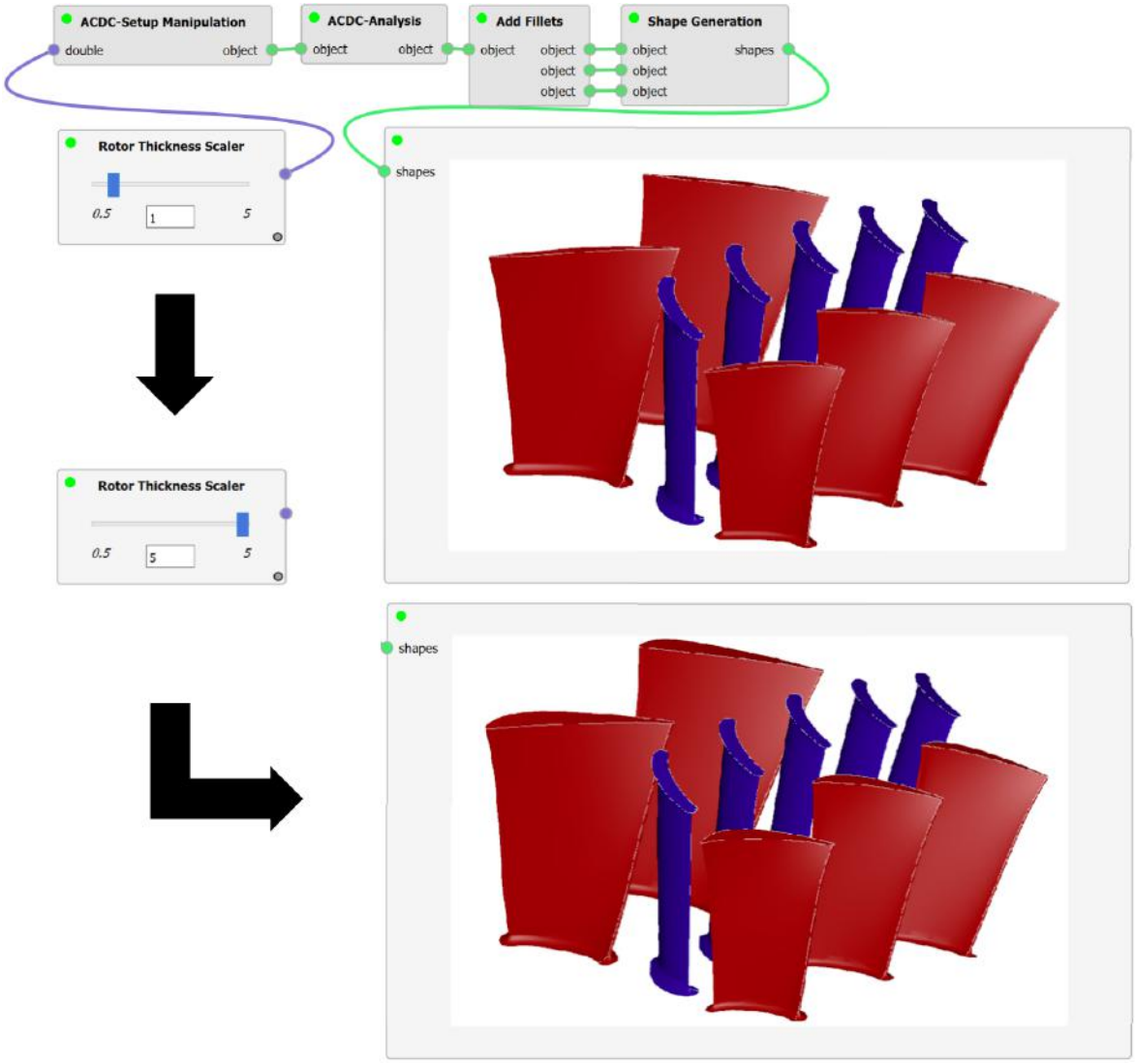

Figure 10 Graph for 3-D geometry design and visualisation based on ACDC design process: Upper graph shows basic process, in the lower part highly increased thickness to chord ratio is applied to rotors.

results into input for 3D CFD simulations.

The focus in this publication is on the processes that are outlined. They are characterised by the ease with which individual process components can be replaced, allowing large parts to be used non-specifically for a wide range of model analyses. In this manner, for instance, the node utilized for the actual aerodynamic analysis or for geometry generation can be substituted with an alternative methodology as needed. A particular strength of the approach is this modularity extended with the scripting capability. In this way, specific changes can be incorporated into the modeling and processing. This also allows seamless linking with other processes or models outside the view of the compressor via the framework. Significant portions of the process chains can be flexibly adapted for other models (e.g., alternative components) or processes. Furthermore, the flexibility of the approach allows it to serve as a valuable foundation for a multitude of potential future developments.

As the overall system allows flexible extensions, more different nodes could be added to the system to add more data manipulation, simulation or result visualization capabilities.

Based on the demonstration of the design of processes in the same framework based on large language models (Reitenbach et al., 2024) an combination with this work could transfer the knowledge of intelligent process design to the framework and support inexperienced users users of the software to build up suitable process chains.

In the future, it will be of particular interest to determine which aspects of the process and the GUI design have most significant influence on engineering design decisions. The analysis of user behaviour will require a study with a large number of users, comparing different graph setups of varying richness and classic non-graph-based user interfaces. Furthermore, it was already mentioned at the beginning that the process in its current design primarily pursues the goal of interactivity. This can only be achieved to a limited extent with longer process steps. A promising approach is to replace computing intensive sub-processes with surrogate models.

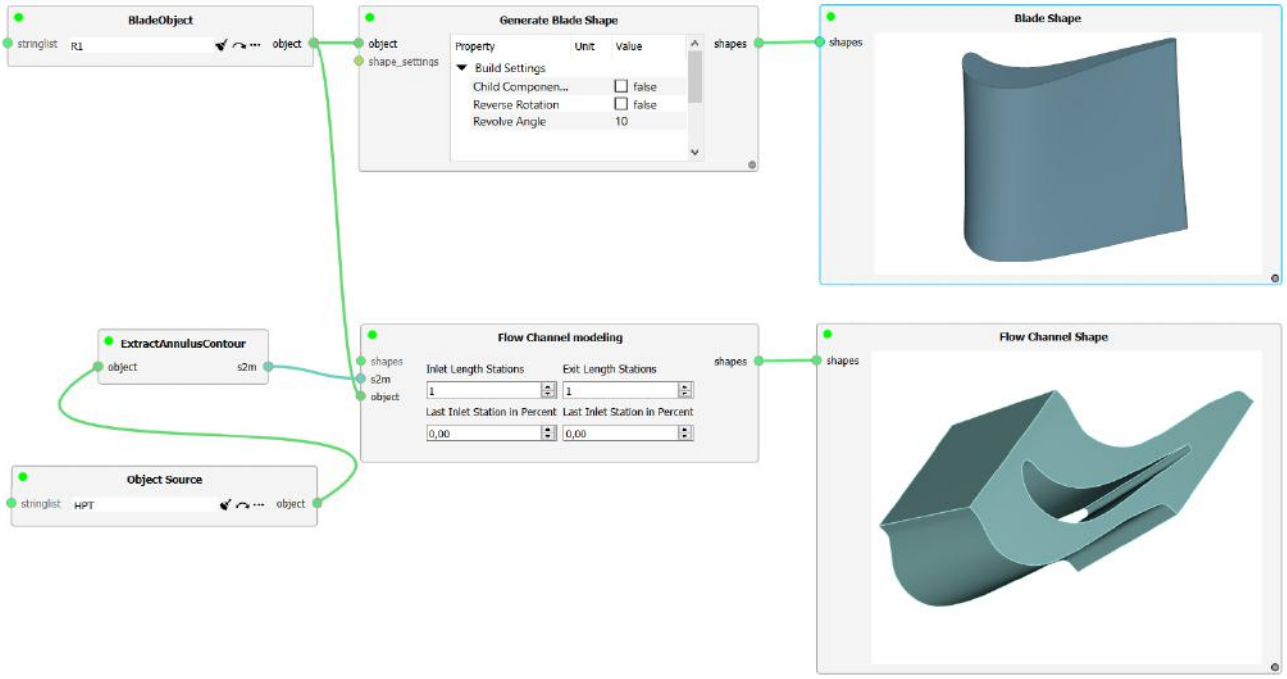

Figure 11 Preparing the blade design geometry for CFD analysis. The fluid volume is generated from the blade shape and the annulus.

References

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Thiel, K. and Wiswedel, B. (2009), 'Knime-the konstanz information miner: version 2.0 and beyond', *AcM SIGKDD explorations Newsletter* **11**(1), 26–31.

Blender Development Team (2022), 'Blender (version 3.1.0) [computer software]', https://blender.org.

Casey, M. and Robinson, C. (2010), 'A new streamline curvature throughflow method for radial turbomachinery'.

Chimakurthi, S. K., Reuss, S., Tooley, M. and Scampoli, S. (2018), 'Ansys workbench system coupling: a state-of-the-art computational framework for analyzing multiphysics problems', *Engineering with Computers* **34**, 385–411.

Concepts NREC (2019), 'Axcent®', https://www.conceptsnrec.com/solutions/software/computer-aided-engineering/detailed-design/axcent.

Dempsey, M. (2006), Dymola for multi-engineering modelling and simulation, *in* '2006 IEEE Vehicle Power and Propulsion Conference', pp. 1–6.

Häßy, J. and Schmeink, J. (2022), Knowledge-based conceptual design methods for geometry and mass estimation of rubber aero engines, *in* '33th Congress of the International Council of the Aeronautical Sciences, ICAS 2022'.

Hinz, M. (2012), *Neue Parametrisierungsstrategien und Methoden der Prozessbeschleunigung für die Verdichteroptimierung*, Shaker Aachen.

Keskin, A. (2006), *Process integration and automated multi-objective optimization supporting aerodynamic compressor design*, BTU Cottbus-Senftenberg.

Maier, M. (2011), Elise 3d-a database-driven engineering and design tool, *in* 'DS 68-9: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 9: Design Methods and Tools pt. 1, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011'.

Mattsson, S. E. and Elmqvist, H. (1997), 'Modelica - an international effort to design the next generation modeling language', *IFAC Proceedings Volumes* **30**(4), 151–155. 7th IFAC Symposium on Computer Aided Control Systems Design (CACSD '97), Gent, Belgium, 28-30 April.
URL: *https://www.sciencedirect.com/science/article/pii/S1474667017436287*

McNeel, R. et al. (2014), 'Grasshopper 3d', *URL: https://www. grasshopper3d. com/3* .

Modelon (2020), 'Modelon impact', https://www.modelon.com/modelon-impact.

Pinaev, D. e. a. (2017), 'Qtnodes. node editor', https://github.com/paceholder/nodeeditor.

Raymond, E. S. (2003), *The art of Unix programming*, Addison-Wesley Professional. ISBN 978-0-1314-2901-7.

Reitenbach, S., Becker, R., Hollmann, C., Wolters, F., Vieweg, M., Schmeink, J., Otten, T. and Siggel, M. (2020), 'Collaborative Aircraft Engine Preliminary Design using a Virtual Engine Platform, Part A: Architecture and Methodology', *AIAA SciTech Forum* .

Reitenbach, S., Hollmann, C., Schmeink, J., Vieweg, M., Otten, T., Haessy, J. and Siggel, M. (2021), Parametric datamodel for collaborative preliminary aircraft engine design, *in* 'AIAA Scitech 2021 Forum'.

Reitenbach, S., Schmeink, J., Bröcker, M., Nöthen, M. and Siggel, M. (2023), 'GTlab — a C++ framework for collaborative engineering', https://github.com/dlr-gtlab/.

Reitenbach, S., Siggel, M. and Bolemant, M. (2024), Enhanced workflow management using an artificial intelligence chatbot, *in* 'AIAA SCITECH 2024 Forum', p. 0917.

Richardson, C., Rymer, J. R., Mines, C., Cullen, A. and Whittaker, D. (2014), 'New development platforms emerge for customer-facing applications', *Forrester: Cambridge, MA, USA* **15**.

Romanenko, L., Moroz, L., Pagur, P., Govoruschenko, Y., Spano, E. and Bertini, F. (2008), 'Advanced gas turbine concept, design and evaluation methodology. preliminary design of highly loaded low pressure gas turbine of aircraft engine', *International Journal of Gas Turbine, Propulsion and Power Systems* **2**(1), 17–23.

Rothrock, L., Barron, K., Simpson, T. W., Frecker, M., Ligetti, C. and Barton, R. R. (2006), 'Applying the proximity compatibility and the control-display compatibility principles to engineering design interfaces', *Human Factors and Ergonomics in Manufacturing & Service Industries* **16**(1), 61–81.

Schnoes, M. and Nicke, E. (2017), 'A database of optimal airfoils for axial compressor throughflow design', *Journal of turbomachinery* **139**(5), 051008.

Siggel, M., Reitenbach, S., Banovic, M. and Pahs, A. (2024), Closing the gap between data model driven geometry generation and hifi cad for automatized and consistent design analysis, *in* 'AIAA SCITECH 2024 Forum', p. 2897.

Simpson, T. W., Barron, K., Rothrock, L., Frecker, M., Barton, R. R. and Ligetti, C. (2007), 'Impact of response delay and training on user performance with text-based and graphical user interfaces for engineering design', *Research in Engineering Design* **18**, 49–65.

Wickens, C. D. and Carswell, C. M. (1995), 'The proximity compatibility principle: Its psychological foundation and relevance to display design', *Human Factors* **37**(3), 473–494.

Wright, P., Miller, D. and Ltd, R.-R. (1991), *An Improved Compressor Performance Prediction Model*, Rolls-Royce plc. **URL:** *https://books.google.de/books?id=NJYlMwEACAAJ*

Zangeneh, M. (2010), Choice of optimum blade loading in application of 3d inverse design to design of pumps and fans.