Simplification and automated modification of existing Isight-processes

D. Otto*, H. Wenzel**

* Department for Design System Engineering, Rolls-Royce Deutschland Ltd & Co KG
** Engineous Software GmbH

Currently, the typical Isight user is a specialist in process integration, optimization and creation of complex processes. He can create complex processes with different external programmes quickly and has a good knowledge of data structure, interfaces and useful control scripts. If some problems appear, the expert user with his experience can react and solve the problem very quickly. By contrast, non-expert users need much more time to build or customise the same complex processes and to solve occurring problems. Therefore, there is a clear barrier for non-expert users to work on complex processes with Isight. There are a number of ways to remove this barrier. One possibility is to simplify the creation process. Therefore, the expert users should create useful components and libraries of subroutines for Isight containing a lot of process knowledge and programme functionality. Non-expert users can then use such components to create processes which generate the same solutions as the reference processes. In addition to the simplification a speed-up of the process can be achieved. Another way is to create templates for complex Isight processes and combine these with another Isight process, which generates all necessary scripts and data structure automatically. The non-expert users manage the modification of a given Isight process via an initialisation file and several control-files. With this approach the complexity of a given Isight process will be achieved with reduced operating errors. This paper demonstrates the above mentioned approaches on two typical industrial design processes. It shows the advantages to program components in Isight and the possibility to modify and control Isight processes in batch.

Keywords: Process Automation, Isight, Isight Components, Turbomachinery

1. Introduction

Design of modern aero engines is typically split into component design tasks related to compressor, combustor and turbine, where each design task is divided again into disciplinary subtasks regarding aerodynamic, design and mechanical aspects, respectively. Increased performance requirements of the customers and call for decreased development time, costs and overhaul time, however, require new design concepts integrating analysis tools from different disciplines. Therefore process integration and automation as well as optimisation are very important issues to meet those requirements. In the following, two approaches for simplification as well as the easy handling of complex processes for non-expert users will be presented. As examples, two typical parts of the compressor blade design process are chosen: the one-dimensional automated meanline prediction and the multidisciplinary automated blading process. Both processes are different in respect of calculation speed, complexity and data input/output.

1.1. The current blade design process

The compressor design process contains four aero design steps. Starting from a one-dimensional meanline prediction (step one), where global flow and geometry parameters for a useful annulus geometry are determined, the subsequent throughflow calculation (step 2) provides radial parameter distributions. Based on these, the blade-to-blade calculation (step 3) is applied to design 2-dimensional blade geometries which fulfil the given flow angles and flow conditions on several radially distributed stream surfaces, where a 2D CFD solver is used. Firstly, 3-dimensional blade geometry is generated by stacking the 2D blade profiles along a specific stacking line, resulting in a spatial free-form surface tangential to all blade profile sections. At the end of the aerodynamic design part a final aerofoil according to additional design rules such as bow and lean is created by means of a 3-dimensional CFD analysis (step 4). If the 3D aerofoil meets the aerodynamic requirements, it is transferred to a CAD system as a non-parameterised object. Otherwise the aerodynamic part of the design process starts again. After the transfer of the aerofoil into the CAD system a

blade is generated through connection of parameterised platform with root, aerofoil and fillet. This blade is finally transferred to a FE system for several structural analyses. Additionally, gas loads on the aerofoil are required as pressure maps which can be calculated with a 2-dimensional or 3-dimensional CFD solver.

2. Simplification and speed up via new Isight-components

One approach to simplify the construction of complex Isight processes is to work with specific self-made components. Such components can contain special calculation routines, data mapping, control of special programmes, and programmes bundles, and can call some dynamic link libraries (so called "DLL") etc. The latter are very powerful to create direct connections to external programmes via a given *C++* interface whereby time consuming programme calls will be sped up compared to the typical programme call in Isight. With components, special input-output GUI`s (graphical user interfaces) can be created and complex Isight JAVA-script components can be replaced. Such options in Isight are chosen to create a bundle of components to speed up and to simplify the generation of standard processes as well as to solve the memory leak problem which can be produce by JAVA-scripts components. The application of self-made components is demonstrated in the following using the example of the automated meanline prediction calculation.

2.1. The automated meanline prediction

The meanline prediction calculation is the first step for a new aerodynamic compressor design where typically an existing compressor design is taken as starting point. Global parameters are annulus line, pressure ratio and number of stages. The goal is to find design parameters and an annulus geometry which fulfil the compressor performance requirements for the aerodynamic design point and several off-design cases. Blockage and surge margin are also outcomes of the meanline calculation. Meanline prediction uses only a one-dimensional model for a complex 3-dimensional flow field. For a better understanding the definitions of the S1 and S2 planes which are proposed by Wu (1952) are helpful, Figure 1. The meanline prediction is a calculation on the S2-plane (by contrast the blade-to-blade calculation is done in the S1-surface).

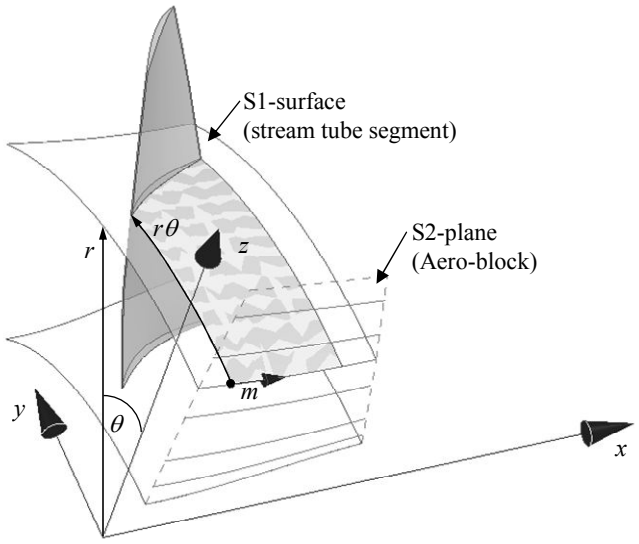

Figure 1. Overview of coordinate systems and calculation planes

For an automated meanline calculation, a clever parameterisation must be used to reduce the number of existing design variables. A proper parameterisation minimises the design parameters but keeps the design freedom at a maximum. Parameter reduction is required to save calculation time (and costs) and to

guarantee the feasibility of the obtained design. With reduction it is possible to automatically run an optimisation in a shorter time period than without. One possibility for parameter reduction with respect to the annulus geometry is the method proposed by Keskin (2006). With such an approach the annulus geometry is determined by two Bézier-Spline curves (Piegl, 1997) for the annulus mid-height line and the annulus thickness distribution, which implies smooth and continues properties of the parameter distribution, as shown in Figure 2.

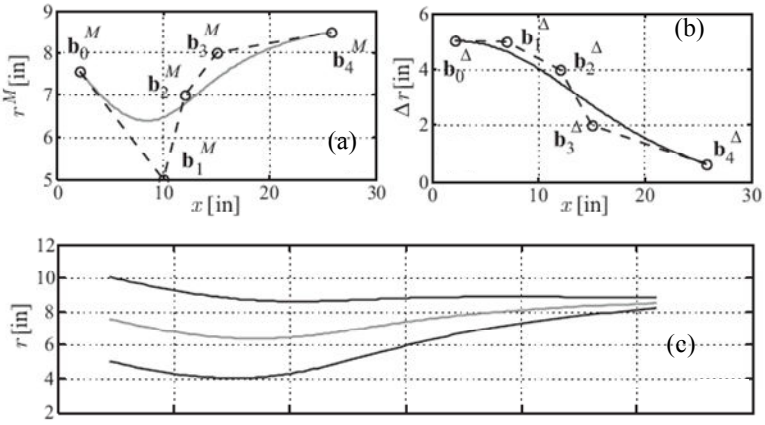

Figure 2. Annulus geometry definition by (a) annulus mid-height line and (b) annulus thickness, each with five control points (labeled with the bold character "b"), as well as (c) resulting annulus geometry (source: Keskin (2007))

Figure 3 shows how the Rolls-Royce meanline prediction process is integrated in Isight. The whole process can be split into the process flow, the design evaluation, and the external programs used within the process. The automated engineering work flow starts with an initialization where required model and design parameters are prepared for the following optimization procedure.

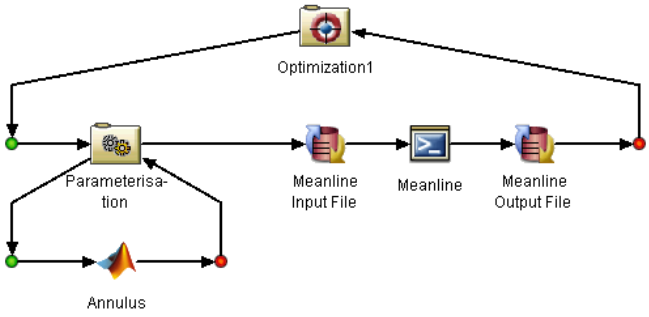

Figure 3. Meanline prediction process build up in *Isight*

If the whole analysis process is done manually, many iterations may be necessary and the design loop has to be run several times with different adjusted design parameters. In the current integrated approach (which includes *MATLAB* for the Bézier-Spline curves generation), the process flow has to be run only once, while the optimization routine interacts with the design evaluation process several times in order to find an

optimal design. The process flow as well as the design evaluation flow are defined and controlled by Isight. When the optimization algorithm requires a new design evaluation, a MATLAB script is started automatically by Isight via the given MATLAB interface which calculates the new annulus geometry based on the new design parameters. This information is transferred into the meanline prediction input file and the meanline prediction programme is invoked through Isight via an OS-command component. After the calculation has converged, the results are bundled in an output file, where Isight extracts basic information required for further parameter evaluation as well as to check if criteria and constraints are fulfilled (Keskin, 2007).

2.2. Isight process by using new components

The described process has two weaknesses: the Bézier-Spline curves generation is done in a commercial code (MATLAB) and the invoking of the external programmes is done via OS-command or a slow internal interface (both ways are slow compared to a direct DLL link). Both weaknesses are eliminated through special Isight components, as shown in Figure 4.

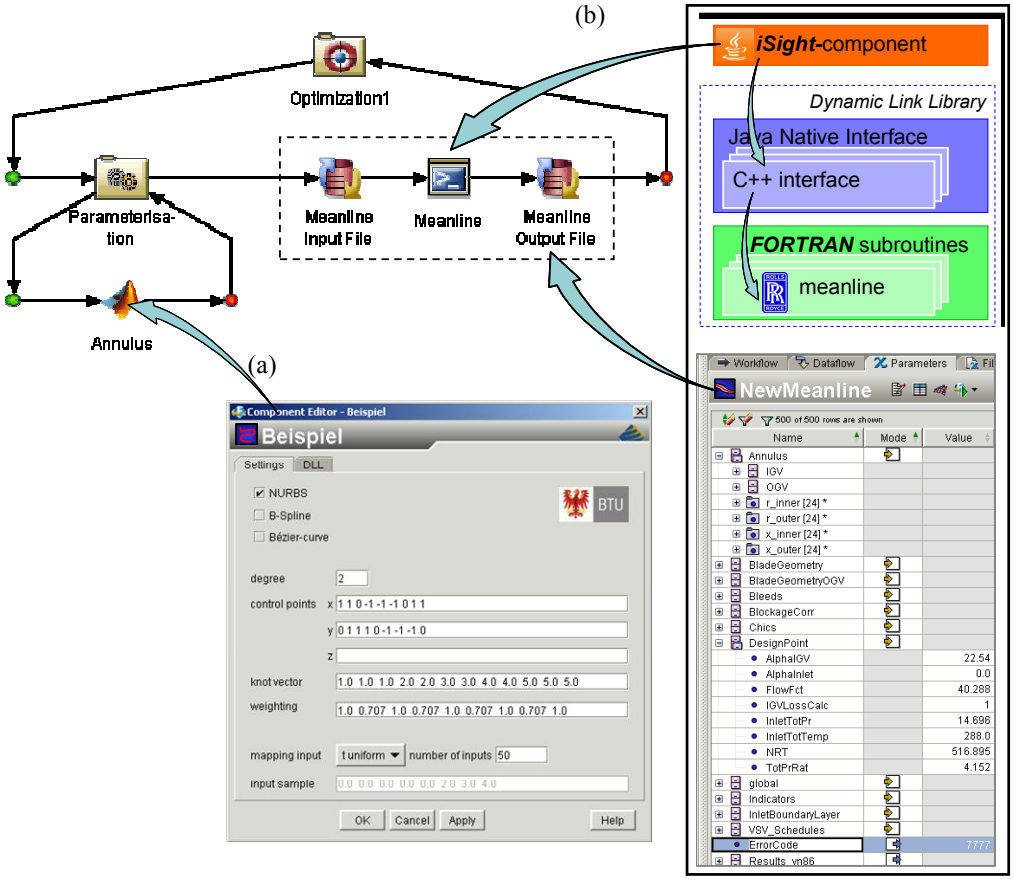

Figure 4. Modification of the given meanline process through (a) NURBS component and (b) "Meanline" component

The Spline generation, normally done in the commercial code MATLAB, is substituted a self-made Isight component, which is called "NURBS." This component support Bézier-Spline curves, B-Splines and NURBS (non-uniform rational basis splines). The calculation of the curves (or curve points) from the input control points is done via *C++* routines (linked via DLL). The GUI of the component supports consistent

entries: control of degree vs. number of control points, knot vector (for B-Splines and NURBS) and weight vector (for NURBS), Figure 4 (a). Both two dimensional and three dimensional curves are available. With this component the license problems with MATLAB are avoided and a significant speed-up compared to MATLAB is achieved.

The invoking of the external programmes via OS-command and the input-output data transfer via the Isight Data Exchanger is slow compared to a direct DLL link. The meanline calculation routine, which is controlled via OS-command, is a programme written in FORTRAN. A direct interface to Isight with the given Isight components is not possible. Also the existing JAVA script component can't solve the problem directly. Therefore a written *C++* routine is used in the JAVA environment whereby an interface to FORTRAN is possible. With this JAVA-FORTRAN interface, linked as DLL, a special Isight "Meanline"-component is created, Figure 4 (b). Additionally a JAVA data parser for meanline input file is written and implemented because hundreds of values have to be entered. Furthermore the automatic parameter naming avoids annoying manual mapping.

With the new components the preliminary meanline process is modified by:

- annulus parameterisation via NURBS component,
- Meanline calculation via wrapped FORTRAN routines

Also a significant speed-up of the whole process is achieved. The new NURBS component has a speed-up factor of 13 compared with MATLAB and the FORTRAN wrapping a factor of 10. With this process acceleration 12,000 function evaluations in 30 minutes on a single PC with 3 GHz are possible, Figure 5. This time speed-up is very useful for any kind of optimisation.

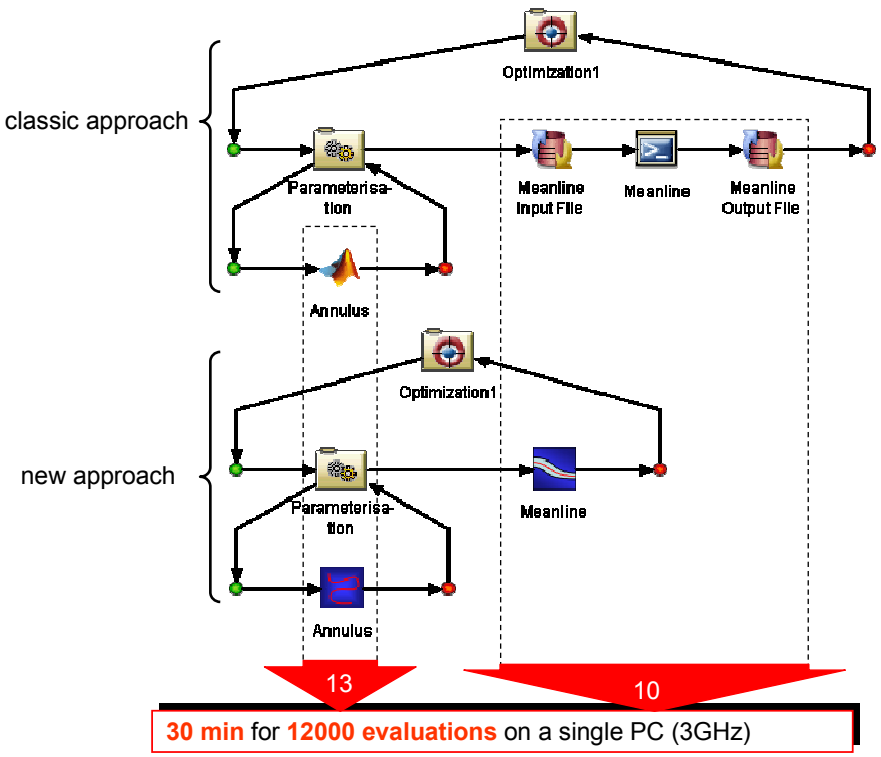

Figure 5. Overview of achieved speed-up factors for both components

3. Generation of standard processes via process templates

Another way to simplify the use and creation of complex Isight processes is described in the following. The main idea is to work with process templates and an initialisation text file. The text file contains all information and data (like specific number of parameters, parameter names, name of used programmes, data structure, etc.) which are necessary to generate a complete working Isight process from the process templates. In the templates the whole structure of the desired process (e.g. mapping of the important data, data flow, etc.) with flexible parameter and data handling is stored. In the shown approach two templates will be used. The first template generates an initialisation process, and the second generates the main process loop (e.g. an optimisation loop). With the initialisation process, all necessary control and system files and scripts will be created. Those files control different programmes (which are needed in the main process loop) in batch and the data flow. The second template generates the main process with the correct parameter names and parameter array sizes as well as correct programme control and data flow. To generate full working Isight processes (based on the templates and the initialisation text file), the templates are manipulated through external JAVA programmes. For manipulation a JAVA programme reads the initialisation file, transfers the data into the templates, and creates the full working Isight process. The described procedure is illustrated with an example. Therefore an automated industrial blade design and optimisation process is used.

3.1. The automated blading process

The intention of the blading process is to find aerofoils which fulfil the performance requirements, aerodynamic and structural criteria. Therefore, the aerofoil is cut with the stream surfaces. At these warped section surfaces, so-called blade-to-blade surfaces S1, intersection lines are created which describe the 2D-blade profiles of the aerofoil in a length-preserving (m-$r\theta$)-coordinate system, Figure 1. These blade profiles can be modified in the blade generator *Parablading* (Gräsel, 2004), which is an in-house tool of Rolls-Royce, via different design parameters like maximum thickness, blade angles, chamber style, etc. *Parablading* has several standard interfaces, like STEP203, for geometry transfer and a special interface to the CAD system *Unigraphics NX*. The latter transfers the aerofoil geometry by using knowledge based rules directly into a CAD kernel whereby the spatial 3-dimensional aerofoil geometry is build up automatically. The advantage of this procedure is that beside the geometry information, the CAD system will also keep the necessary assembly information with the platform, root, and fillet. This results in a fully parametric CAD model which allows design changes on both the aerofoil and the platform.

The shape of the aerofoil can be controlled by descriptive parameters within the blade generation tool *Parablading* for each blade profile. A direct modification of selected profile parameters would lead to non-smooth transitions and undesired leaps between sections. On this account, radial distributions of descriptive parameters are used which are described as parametric curves in 2-dimensional spaces spanned by particular parameter and section number. In the following investigation the desired design parameters of all sections of a blade are used as design quantities where the correlation between section number of an individual section is described by a B-Spline (Piegl, 1997) with given control points. Figure 6 (a) illustrates an example of desired parameter distributions (in this case maximum thickness of the blade profile and the blade inlet angle). Design changes can be performed by horizontal and vertical movements of the control points and extracting the values of the design parameters for a specific section from the B-Spline. An advantage of this parameterization is an overall parameter reduction, since the number of control points is less than the number of blade sections. In addition any design change on control points will result in smooth distributions for the blade design parameters. All B-Spline modifications are done in an external programme which performs adequate modifications on these distributions and extracts the desired discrete thickness and blade inlet angle values for the various blade sections. These values are written to the input file of *Parablading* for aerofoil modification.

After each aerofoil design modification, multiple 2D flow calculations with the two-dimensional CFD solver *Mises* (Drela, 1998) are performed on several radial positions which support the subsequent stress analysis by providing more accurate gas loads to be handled as aerodynamic boundary conditions, and by providing characteristic aerodynamic design values like loss, distribution of the Mach number, or the boundary layer form factor. With a modified blade model from *Unigraphics NX*, where also the fillet radii between aerofoil and platform are design parameters, Figure 6 (b), and with the new generated gas loads, static and dynamic finite element analyses are performed with ANSYS to calculate stresses and

eigenfrequencies. The whole process is fully automated and implemented in Isight with all given standard Isight components. All programmes can be run in batch mode and are controlled by special control files and scripts, so that they can be controlled via Isight (Otto, 2006).

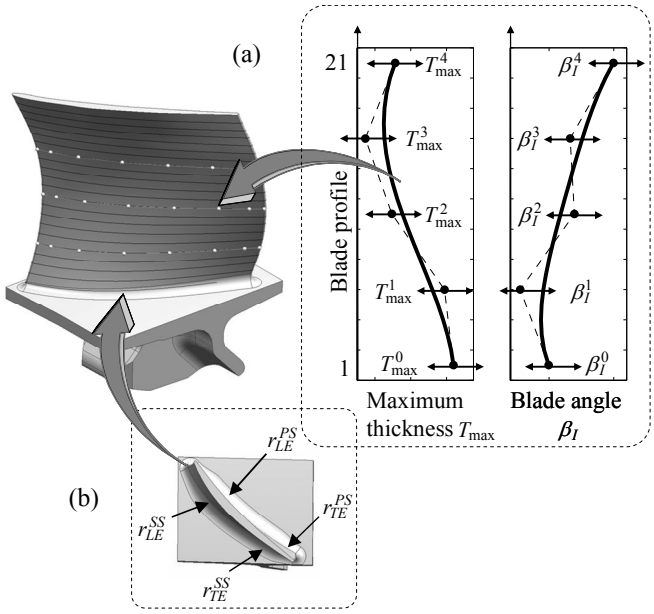

Figure 6. Possible blade design parameters for the (a) airfoil and (b) the fillet

3.2. *Isight* process by templates

The automatic blading process has a lot of external input-output data flow. Also the control scripts are external. Both can be sources of errors, which can cause the process to fail. Additionally one function evaluation (more precisely: one process-loop) is very time consuming, on average 15 minutes with a quad-core HPC (4 GHz and 32 GByte memory). If in the context of an optimisation wrong design parameters are selected in Isight, the whole optimisation is unusable. This is also a source of errors due to the fact that a lot of design parameters are available. Expert users, who build up the blading process in Isight or work with the process very often, have a good knowledge to solve some problems and avoid different errors. By contrast, non-expert users are more prone to make errors and they need a lot of time to build up the same Isight process.

To help non-expert users and to reduce the possibility of errors, the use of Isight process templates is useful. The first step is to fill the initialisation file with important data like directory and data structure, names of desired aerofoil design parameters, and the number of control points for the B-Splines (which represent the radial distribution of the aerofoil design parameters number of eigenmodes (which will be analysed with the FE system), etc. The next step is the generation of the Isight initialisation process: based on the existing template a JAVA programme transfers the data from the initialisation file into the template and creates in this case an Isight initialisation process, Figure 7 (a). This process must be run once. The results are all necessary control files and scripts, and the control point coordinates of the B-Splines achieved through fitting of the desired aerofoil design parameters respectively to the given number of control points. The latter are the aerofoil design parameters which can be modified in the main optimisation loop.

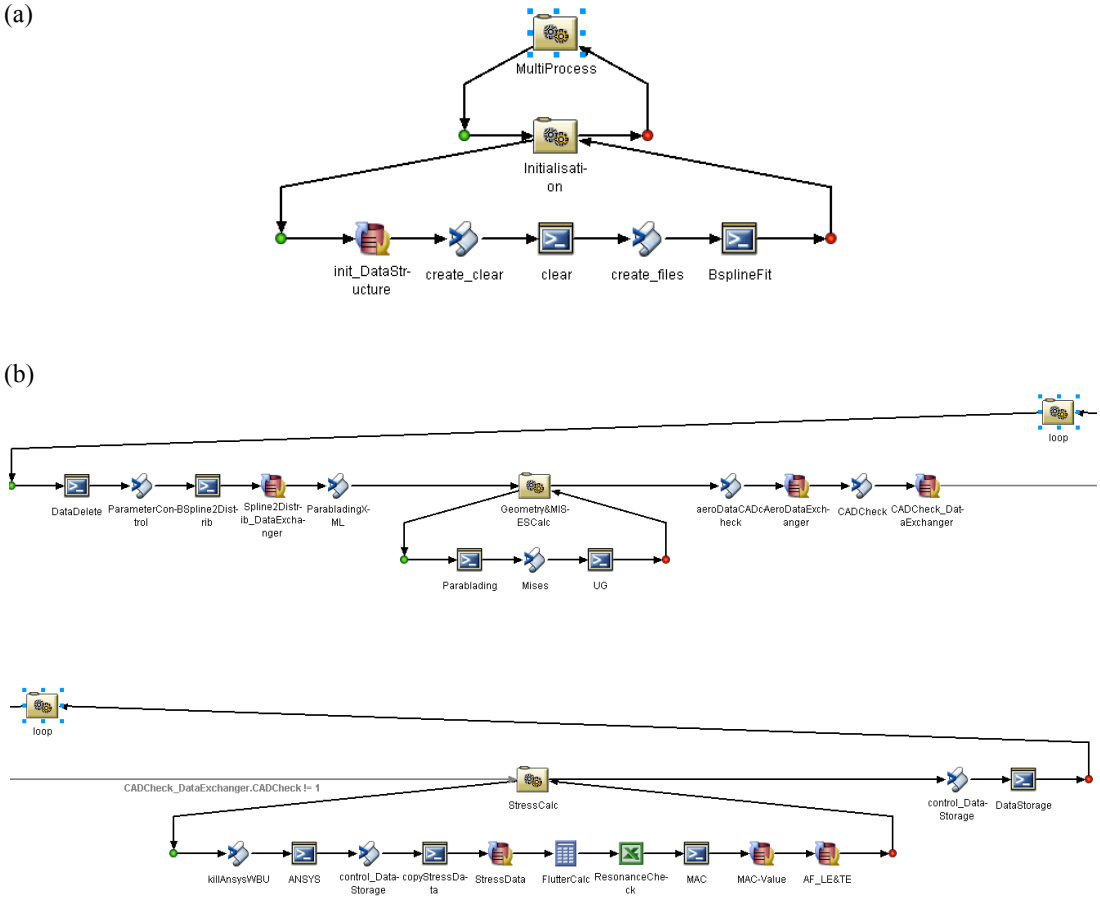

Figure 7. Both Isight processes which will be created from the templates: (a) initialisation process and (b) the main process

The last step is the creation of the main process (also done with a JAVA programme). Therefore the initialisation and the control points coordinates are necessary since the latter are the design parameters and therefore the parameter structure (name and sizing of used arrays, etc.) must be modified. A full working Isight process is the result of the last step, Figure 7 (b). The user can decide what he wants to do like DOE or optimisation, etc. He can select just the design parameters which were given in the initialisation file, Figure 8, and he can set upper and lower boundaries for the parameters as well as constraints and goals for the DOE or optimisation run.

4. Summary

The paper presents two approaches to speed up and simplify the creation of existing Isight processes. For the first approach, self-made Isight components are created by using the dynamic link library and some C++/FORTRAN as well as JAVA routines. With components the process can be speeded up and the parameter handling can be simplified. The second approach works with an initialisation file and Isight templates which contain complex processes. By combining the initialisation file and templates with a JAVA programme, a complete Isight process can be created. The advantage for the latter approach is the reduction of sources of errors. Additionally a good knowledge about the desired process is not necessary to start an optimisation or a design of experiments (DOE) run.

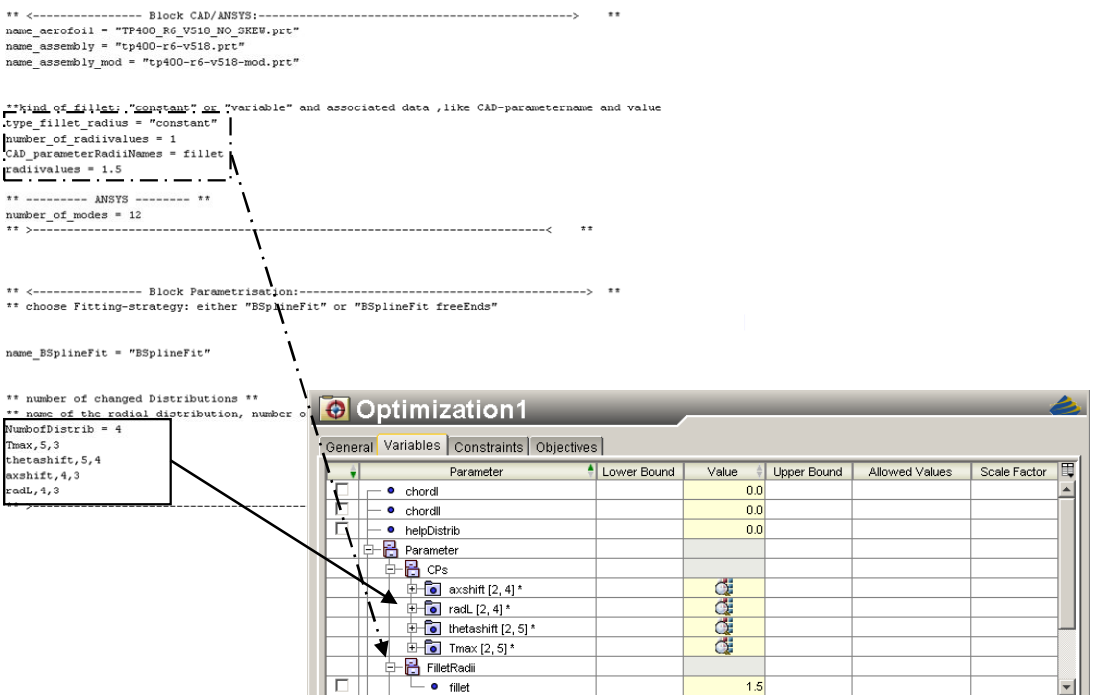

Figure 8. Design parameters in Isight, which are defined in an initialisation file and created in the Isight process via an external JAVA programme

5. REFERENCES

Drela, M. and Youngren, H., "A User's Guide to MISES 2.53," MIT Computational Aerospace Sciences Laboratory, Cambridge, 1998.

Gräsel, J., Keskin, A., Swoboda, M., Przewozny, H. and Saxer A., "A Full Parametric Model for Turbomachinery Blade Design and Optimisation," Proceedings of ASME DETC 2004, DETC2004-57467, 2004.

Keskin, A. and Bestle, D., "Application of Multi-Objective Optimization to Axial Compressor Preliminary Design," Aerospace Science and Technology 2006, vol. 10, no. 7, pp. 581-589, 2006.

Keskin, A., "Process Integration and Automated Multi-Objective Optimization Supporting Aerodynamic Compressor Design," Ph. D. Thesis, Aachen, Shaker, 2007.

Otto, D. and Bestle, D., "Automation and Optimisation of Compressor Blade Design with Respect to Mechanical Criteria," German Aeronautics and Astronautics Aerospace Congress, 2006.

Piegl, L. and Tiller, W., "The NURBS Book," Berlin, Springer, 1997.

Wu, C.-H. "A General Theory of Three-Dimensional Flow in Subsonic and Supersonic Turbomachines of Axial-, Radial-, and Mixed-Flow Types," NACA TN-2604, 1952.