System Integration as Key for Improving and Speeding up the Preliminary Design Phase of Aero Engines

Philipp Kupijai[1], Dieter Bestle[1] and Daniel Kickenweitz[2]

[1] Brandenburg University of Technology Cottbus, [2] Rolls-Royce Deutschland Ltd & Co KG

Abstract: Usually the development of a new engine is initialized by the request of an aircraft manufacturer who formulates basic requirements for the propulsion system. Preliminary engine design is then the first design phase at the aero-engine manufacturer where the engineer's task is to develop a proper engine concept fulfilling all requirements and to respond to the aircraft company in a short time. At this early design stage only simple models can be used, but the decisions made are far-reaching and need to be precise. Therefore, engine component scaling tools are applied to increase prediction accuracy. Accessing these methods, which have been developed by different component specialists, implies high manual effort due to manual data transfer and due to the fact that design is always an iterative process. This paper demonstrates an automated design process utilizing Java based Isight components which are a robust, secure and fast alternative to common Excel sheets and may even be the superior solution. The process will be illustrated by design and optimization studies on two application examples showing the benefits in terms of saved runtime and better solutions than can be obtained by experience driven, human search.

Keywords: aero engine, automation, multi-objective design optimization, process integration, Isight

1. Introduction

Preliminary engine design is a challenging task. Only few information and short time is available but relatively high prediction accuracy expected. In general, a combination of generic methods and knowledge based engineering (KBE) tools, which reflect more realistic effects and increase e.g. efficiency prognoses, are combined. The basic procedure is shown in Figure 1. After receiving the engine requirements from the aircraft company, the engine manufacturer begins with the system level preliminary design by iteratively using KBE tools implemented as Excel sheets and an engine performance program being an industrial in-house code. Data have to be copied manually from one program to the other, again and again for a couple of iterations. After finding an iterated solution, a validation of the data set is performed on component level with tools of higher accuracy. As a result the general arrangement can be drawn for the engine bid and engine weight and costs can be predicted. This procedure is a time consuming approach, especially due to the high manual effort during the system level preliminary design. Therefore, often not much time is left for design studies to really find the best solution. Hence, there exists a strong demand for an automated design and optimization process.

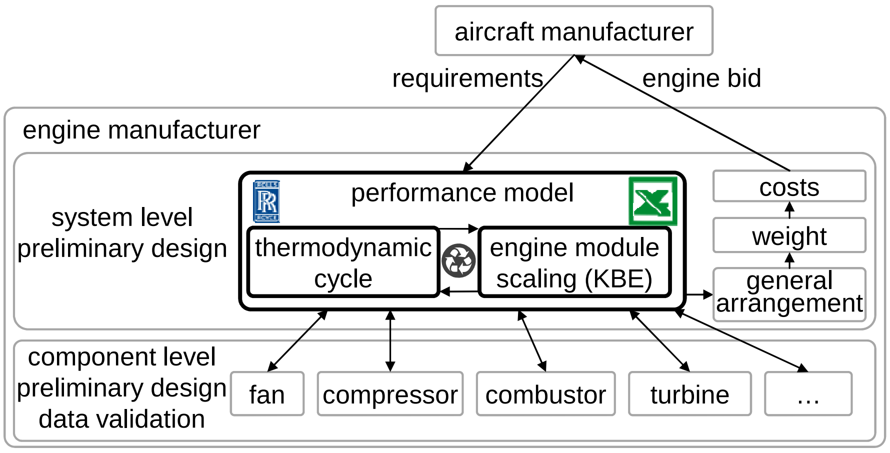

Figure 1. General preliminary engine design process.

This paper is structured in three main parts. Firstly, an assessment of Excel and Java within design processes and their behavior in Isight is given. Then, the generation of the automated design process is explained, finally followed by two application examples of automated design processes.

2. Assessment of Excel and Java Behavior in Isight Processes

There are several approaches to realize an automated design process. Using batch or shell scripting is a rather direct and cheap solution, however, the use of commercial system integration platforms like Isight from Dassault Systemes (2011) are more advantageous due to training and support capabilities by the vendor, graphical user interfaces for easy use and a huge number of functionalities which are already available. It would be possible to take the existing process and just put the components in an Isight workflow. However, due to the author's experience Excel is not the best choice for Isight processes. E.g., Excel execution in Isight can fail if pop-ups are used. Moreover, some Excel VBA macros do not run in Isight. In general, the use of VBA macros in Excel may be critical because not all macros are compatible with different Excel versions, e.g. Excel 2007 does not know all macros from Excel 2003. An update to a later Excel version thus may cause trouble, and a lot of rework could become necessary to solve arising macro problems. Further, a great amount of effort is required to realize version control in Excel. In contrast to common programming languages, to the author's knowledge there is no simple version control tool for Excel available with access to a repository. Especially if no protected cells are used in distributed Excel sheets, a user can change his local copies which can lead to inconsistencies. This problem becomes even worse if Excel sheets are planned to be used as black boxes, e.g. by sub-contractors, and unwanted knowledge transfer has to be avoided. Grossman (2008) suggests a procedure, but this is associated with huge effort. Additionally, the risk of programming errors may be higher in Excel sheets compared to classical programming languages. McConnell (1996) comments that this is not due to Excel itself but due to the, in general, more sloppy way of programming Excel sheets where basic programming rules are neglected. Panko (1998) quantifies in his study the alarming high error rate of Excel sheets used by big companies for their finances and accounting. The fact that Excel cells can be both values and formulas simultaneously can be beneficial but may also increase the risk of errors.

In contrast to Excel, classical programming languages like the script language Java can be managed with version control, handled as black boxes, and they run more stable in Isight. Further it is simpler to comment source code. A common argument of Excel users is that Java requires a higher level of programming skills than Excel, but this is not completely true. In order to generate a professional Excel sheet dealing with the disadvantages mentioned above, high Excel programming skills are necessary. Therefore, the question arises, why not benefit from the advantages of Java programs?

One of the major advantages of Java is the higher computing speed compared to Excel, as the following example illustrates. A simple calculation with the few successive steps

$$z = x \cdot y, \quad z = z^2, \quad z = z + x + y, \quad z = |z|, \quad z = \sqrt{z} \tag{1}$$

has been implemented in an Excel sheet and in a Java based Isight component. Then, two Isight processes have been set up as shown in Figure 2, both varying the input parameters *x* and *y* using a DoE (Design of Experiments). Both processes use the same seed for generating exactly the same parameter values, where two different sample sizes are used in the DoEs: 1000 and 10,000. Both processes are repeated ten times, where the runtime is shown in Figure 3a. Red symbols mark DoEs with 10,000 samples and blue symbols the DoE with 1000 samples. In both studies, Excel (marked with squares) requires much more runtime than Java (marked with circles), respectively. However, also an important fact is that the Excel runtime increases during large DoEs (ordinate is plotted with logarithmic scale!). Additionally the allocated RAM increases from execution to execution if the Isight Design Gateway and Runtime Gateway are not closed after each execution of a DoE. For a comparison between Excel and Isight runtimes, also the speed-up factor

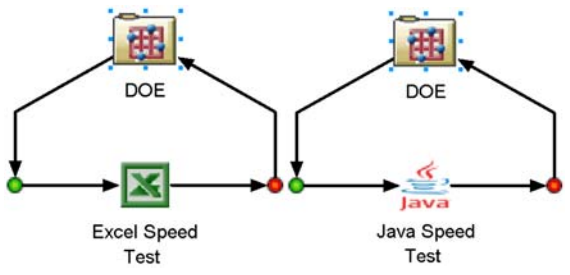

Figure 2. Processes for runtime comparison between Excel and Java in Isight.

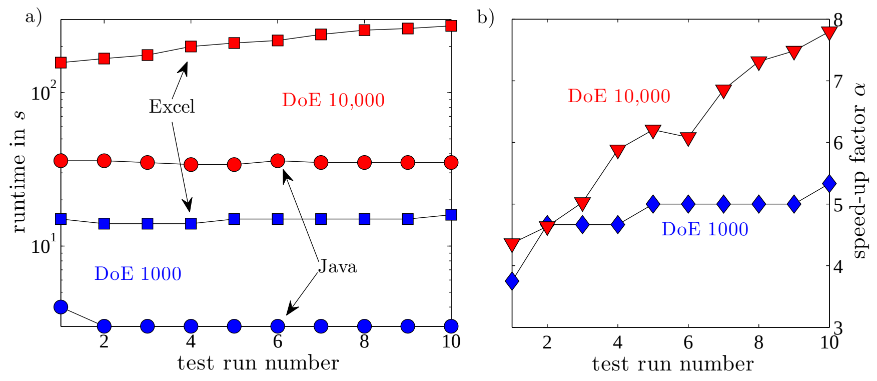

Figure 3. Runtime[1] comparison between Java and Excel in Isight: a) Excel ■ and Java ● runtime for DoEs with 1000 (blue) and 10,000 (red) samples, b) speed-up factor for DoEs with 1000 (blue) and 10,000 (red) samples

$$\alpha = \frac{t_{Excel}}{t_{Java}} \tag{2}$$

is displayed in Figure 3b for each execution. For this simple example the speed-up factor varies between 3.5 and 8. However, Kupijai and Bestle (2011) experienced even speed-up factors up to 300 if Excel files become larger and more complex. Obviously, this speed-up is a good reason for choosing Java instead of Excel, and the effort for transferring existing Excel sheets to Java based Isight components is well justified.

3. Generation of an automated preliminary engine design process

The initial industrial design process for aero engines accesses a series of Excel sheets which had to be replaced by Java components. The major effort during setting up the automated design process has been turned out to be the conversion procedure. Although theoretically well known that a structured programming approach is highly beneficial, in practice it looks different due to strong time limits and the pressure on the programmers to deliver results. Figure 4 illustrates the different steps during the conversion phase. Firstly, the source code (Excel/VBA) has been analyzed and the methods extracted. This step is very useful for finding and correcting errors, and the methods themselves can be improved. Afterwards program architecture can be developed, where common methods, like numerical or thermodynamical methods, wherever possible are extracted to external

[1] Intel® Core™2 Duo CPU P8700 @ 2.53GHz, 2GB RAM

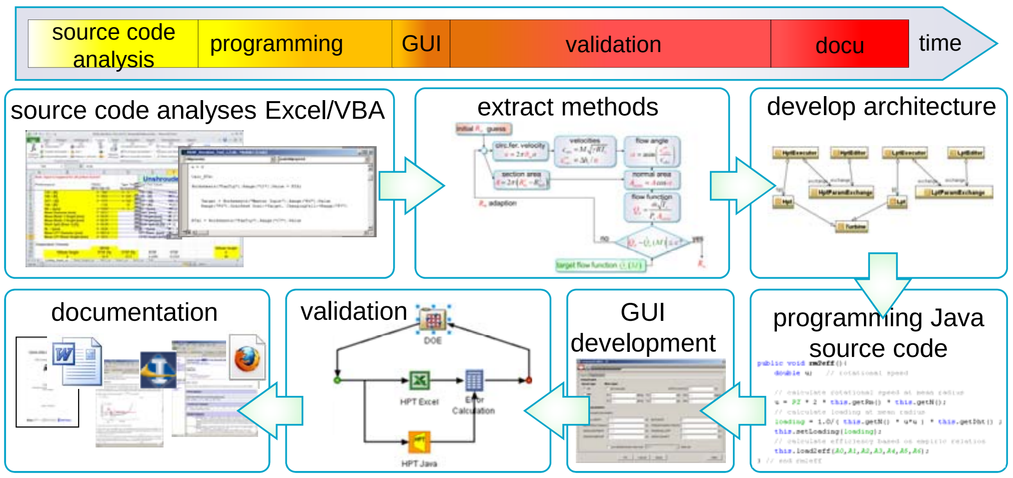

Figure 4. Phases of conversion procedure from Excel sheets into Java based Isight components based on classical programming development approach and estimation of qualitative amount of phase time compared to overall time.

classes for reuse or use and existing in-house or open source classes are used. Only then the (Java) source code should be written and the graphical user interface implemented. Dassault Systemes (2011) already provides the interfaces between the new Java program and Isight (*sdk.runtime.Component*, *desktop.sdk.DesktopEditor*, etc.).

Finally, two very important phases follow which are often neglected or handled without the required care: Program and methods have to be validated and documented. The upper part of Figure 4 shows a qualitative time bar relating the effort for different development phases based on the author's experience. Validation and documentation require about the same time and effort like the previous phases. For a simple conversion without any change on the methods, the validation has been performed within an Isight process where a DoE varies the design parameters, Excel and Isight are executed in parallel, and the results are compared to each other. Documentations have been provided at different levels: HTML Java docs in Java source code, Word and PDF manuals, and HTML user help included in the Isight component.

A general goal is to create a robust design process, where robustness refers to stable running. Another aspect is a meaningful error handling as shown in Figure 5. The approach tries to avoid termination of processes wherever possible. If the design process or optimization consists of a high number of evaluations, it has to be decided if an error incidence compromises the general process execution or only the current evaluation, e.g. if a parameter is out of allowed range. If the error is not fatal, it has to be checked whether the error can be corrected by the system itself and how a warning should be communicated to the user. Therefore, it may be beneficial to abstain from calling exceptions which lead to a process termination wherever possible.

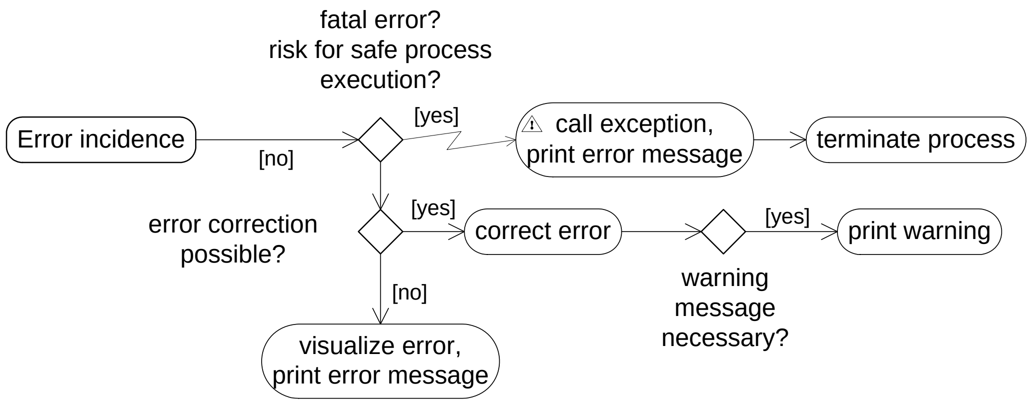

Figure 5. General error handling within KBE Isight components.

The KBE tools have been implemented as Java based Isight components. Further an interface to the in-house performance program has been established. All components have been integrated to an automated design process called *Computational Preliminary Power-Plant Optimization* (C3PO). The design evaluation within C3PO follows the process flow shown in the activity diagram in Figure 6. After the initialization of the design process, the iteration between the performance calculation and the engine component scaling starts. The performance calculation involves the design point calculation and a series of off-design calculations. The engine component scaling is performed separately for the different engine components, like intake, fan, engine section stator (ESS), fan OGV, booster, HPC, combustor, HPT, LPT, cooling flows for the turbine, and some additional process parameters, respectively. Mainly those components predict efficiencies, basic geometry and architectural properties like number of stages etc. After each iteration some post processing is performed checking the convergence and whether errors occurred or not. Finally, if the iteration has finished, engine mass is predicted. The corresponding Isight workflow is shown in Figure 7 where only the part for a single engine evaluation is shown. The C3PO task can of course be integrated into different super processes with different kind of drivers, like DoE or optimizations. Two application examples are given in the next section.

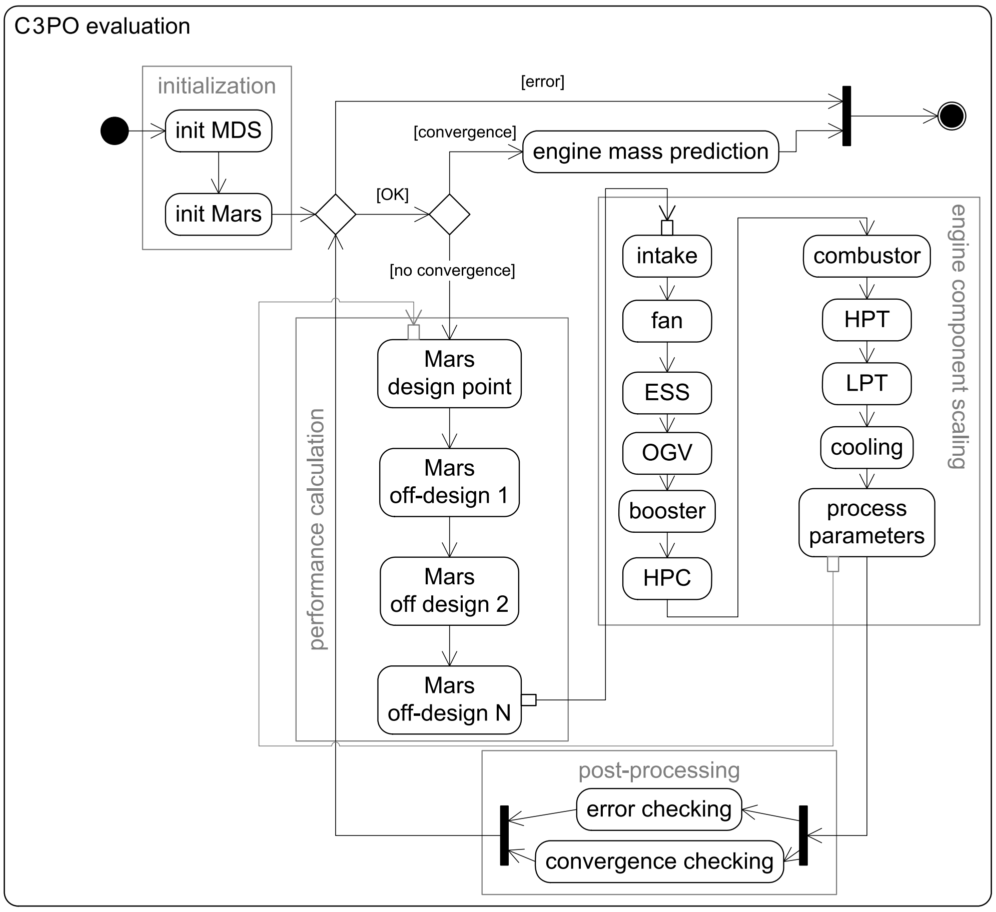

Figure 6. Activity diagram for C3PO engine evaluation using the Mars performance program and Rolls-Royce Master Data Set (MDS) for data management.

4. Application examples of the automated design process

In the following, two different application examples are shown in order to demonstrate the benefit of the proper automated design process. The first example is a design study within a European Project called Crescendo (2009-2012), where a coupled collaborative aircraft engine optimization has been performed by different partners. The idea of this approach is to integrate the engine model by the engine manufacturer as so called "*rubber*" engine within the aircraft manufacturer's optimization process. Lammen et al. (2013) explain in detail the collaborative optimization approach. The rubber engine consists of a surrogate model, Forrester et al. (2008), which is built

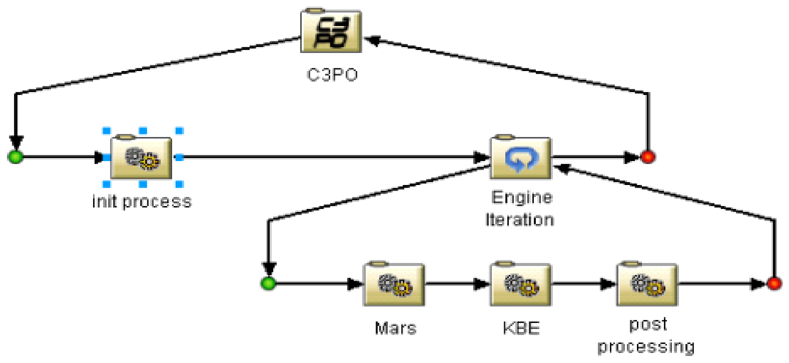

Figure 7. Automated design process C3PO in Isight

on a data set consisting of 1125 engine configurations in a first collaborative design process and on 400 configurations in a second loop, respectively. However, this paper only reports on the engine design part of the first iteration.

The design parameter vector

$$\mathbf{p} = \begin{bmatrix} F_N^{MTO} & F_N^{MCL} & T_3 & T_{405} & D_F \end{bmatrix}^{\mathrm{T}} \qquad \textbf{(3)}$$

summarizes net thrust at maximum the take-off F_N^{MTO} and maximum climb F_N^{MCL} thrust flight cases which are of importance for the aircraft manufacturer, as well as temperatures at compressor outlet T_3 and turbine inlet T_{405}, which have impact on the engine core, as well as the fan diameter D_F. Figure 8 shows results of a design study where specific fuel consumption is plotted against engine weight. Minimizations of these two quantities are two general design objectives for aero engines.

Besides the fact that this design study could not have been performed without the automated design process, the following consideration may compare the required runtime with the estimated runtime that would have been necessary for the manual approach. Let T_N be net runtime of the process which only depends on the number N_D of evaluated designs and the mean evaluation time for a single design t_N, i.e. $T_N = N_D \cdot t_N$. If the net time exceeds the time of a working day $t_d = 8h$ or even a working week $t_W = 40h$, the breaks between days $t_{BD} = 16h$ and weeks $t_{BWE} = 48h$ have to be considered. Therefore, a crude estimation of required gross time for an manual process may be given as

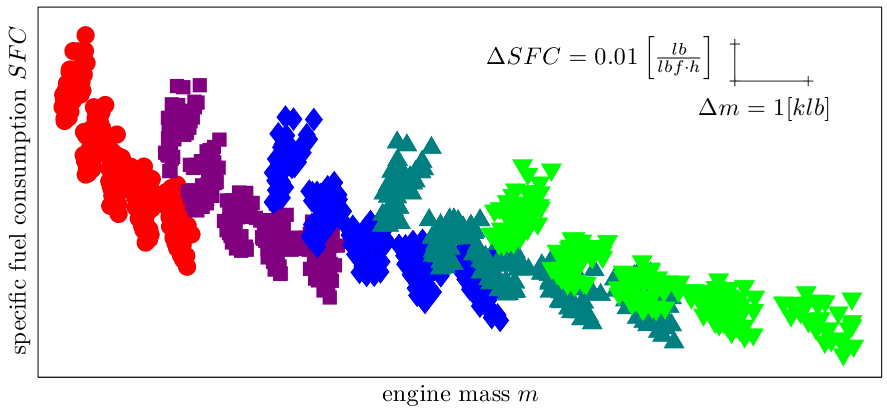

Figure 8. Results from engine design study where each color refers to a different thrust class.

$$T = T_N + \left\lfloor \frac{T_N}{t_d} \right\rfloor \cdot t_{BD} + \left\lfloor \frac{T_N}{t_W} \right\rfloor \cdot t_{BWE} \,. \tag{4}$$

This is, of course, only a simplified equation neglecting effects like holidays, illness, additional work tasks, tiring of the engineer, and assuming a constant mean evaluation time $t_N = \text{const}$. A comparison of this estimated runtime of the manual approach with actual C3PO runtime yields a total process speed-up factor of 25. If only the engineer's time for initializing the process is considered the speed up factor is even 300.

The second application example deals with a multi-objective design optimization. In accordance to Figure 8, the goals are to minimize specific fuel consumption SFC and engine mass m:

$$\min_{\mathbf{p}\in\mathbb{R}^3} \left[SFC(\mathbf{p}) \quad m(\mathbf{p}) \right]^{\mathrm{T}} \tag{5}$$

where the design vector $\mathbf{p} = \left[D_F \quad T_3 \quad T_{405} \right] \in \mathbb{R}^3$ consists of fan diameter D_F, compressor outlet temperature T_3, and HPT stator outlet temperature T_{405}. The goal of this bi-criterion design problem is to find a proper set of non-dominated designs in the shortest time. A possible approach is to use multi-objective genetic algorithms (MOGA) which have the advantages of finding multiple Pareto-optimal designs in a single sweep, reducing the risk of getting trapped in local minima of multimodal objective functions and being robust against noisy or non-existing design evaluations which occur in case of inconsistent data sets. Isight offers several such optimization algorithms. The following optimization study uses the Archive-based Micro Genetic Algorithm AMGA introduced by Tiwari et al. (2008), because the algorithm showed best performance.

In order to further speed-up the optimization process, different approaches have been implemented, where one approach is the use of response surface models

$$f(\mathbf{p}) = \hat{f}(\mathbf{p},\mathbf{b}) + \varepsilon_f(\mathbf{p},\mathbf{b}),\ \mathbf{p} \in \mathbb{R}^n\,. \tag{6}$$

Here the original model behavior $f(\mathbf{p})$ is approximated by a simplified mathematical model $\hat{f}(\mathbf{p},\mathbf{b})$ depending on the design vector $\mathbf{p}$ and some model parameters $\mathbf{b}$ which are adapted to the specific model functions; $\varepsilon_f(\mathbf{p},\mathbf{b})$ is the error between the original model and the response surface model. Further, the problem formulation in Equation 5 has been scalarized as

$$\min_{\mathbf{p}\in\mathbb{R}^3}\left(\hat{SFC}(\mathbf{p},\mathbf{b})\right) \text{ s.t. } \hat{m}(\mathbf{p},\mathbf{b}) \le m^c \tag{7}$$

where $\hat{SFC}$ is the response model for SFC and $\hat{m}$ for engine mass. The idea of the scalarization is to reduce the problem dimension to a single objective minimization problem and to use the second objective as constraint $\hat{m} \le m^c$ with moving the bounds m^c, i.e. the optimization is repeated with different constraint settings in order to move along the Pareto front. The advantage of this approach is that for scalar optimization problems more efficient algorithms are available, e.g. Covariance Matrix Adaption Evolution Strategy CMA-ES introduced by Hanson (2006) which has been implemented in Isight. Flassig and Swoboda (2011) demonstrate the robust performance of this algorithm. More details about the surrogate model based engine optimization study are given by Kupijai et al. (2012).

Some results are shown in Figure 9 where the dashed line shows results from the AMGA

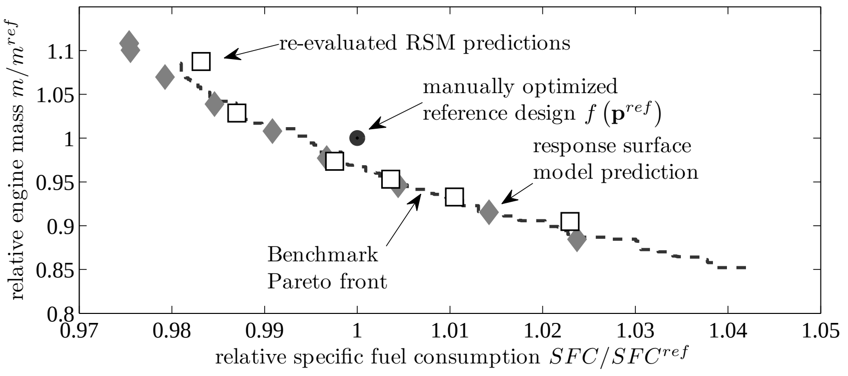

Figure 9. Optimization results based on response surface model and reduced problem formulation.

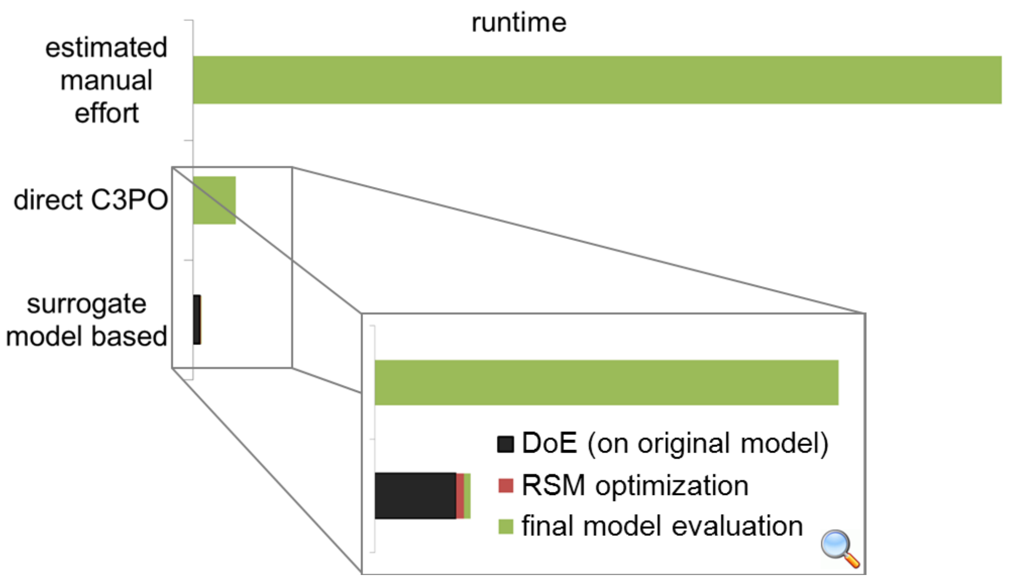

Figure 10. Runtime comparison for different optimization approaches.

optimization with the original model and grey diamonds illustrate optimal designs predicted by the surrogate model. However, in order to trust the surrogate model predictions, those designs have to be re-evaluated with the original evaluation process. The results of this re-evaluation are marked by white squares demonstrating that the surrogate model provides comparable results with the those of the original model.

However, the runtime, as displayed in Figure 10, shows the great benefit of the approach. The time reduction compared to the manual approach according to estimation in Equation 4 reveals a speed-up factor of about 19 when using the automated process C3PO and about 97 when additional acceleration approaches like problem scalarization and surrogate models are used.

5. Conclusion

The paper briefly describes the generation of an automated Isight based design process for flight engine preliminary design. Excel and Java as potential evaluation tools in automated design processes are compared and discussed where clear advantages of Java are pointed out. Different steps for the conversion procedure from Excel to Java are identified. The automated design process enables to run large design studies and multi-objective optimization. Two example applications of the automated design process C3PO are shown where the benefits of the automated design process become obvious.

6. Acknowledgment

The research has been carried out within the CRESCENDO Integrated Project, which is partly sponsored by the European Union’s Seventh Framework Program (FP7/2007-2013) under grant agreement No. 234344 (www.crescendo-fp7.eu). In addition, support by Rolls-Royce Deutschland is greatly acknowledged.

7. References

1. CRESCENDO (2009-2012). “Collaborative Robust Engineering using Simulation Capabilities Enabling Next Design Optimisation”. European Union’s Seventh Framework Program. URL: www.crescendo-fp7.eu.
2. Dassault Systemes (2011). Isight Version 5.5, Development Guide.
3. Flassig, P. and M. Swoboda (2011). “Current Isight Applications Supporting Optimal Aerodynamic Compressor Design”. In: Proceedings of ADOS - Aerodynamic Design Optimisation Seminar, Derby.
4. Forrester, A., A. Sobester and A. Keane (2008). “Engineering Design via Surrogate Modelling: A Practical Guide”. Chichester: John Wiley and Sons.
5. Grossman, T. A. (2008). “Source Code Protection for Applications Written in Microsoft Excel and Google Spreadsheet”. CoRR (4774). abs/0801.4774.
6. Hansen, N. (2006). “The CMA Evolution Strategy: A Comparing Review”. Stud-Fuzz 192, 75–102.
7. Kupijai, P., and Bestle, D., 2011. “Beschleunigung der Triebwerksvorauslegung durch Verwendung problemangepasster Isight-Komponenten”. In: Proceedings of the Deutsche SIMULIA-Konferenz. Bamberg.
8. Kupijai, P., D. Bestle, P. Flassig and D. Kickenweitz (2012). “Automated Multi-Objective Optimization Process for Preliminary Engine Design”. In: Proceedings of ASME Turbo Expo, Copenhagen, GT2012-68612.
9. Lammen, W., D. Kickenweitz, T. Laudan and P. Kupijai (2012). „Integrate Engine Manufacturer’s Knowledge into the Preliminary Aircraft Sizing Process”. To be published in Proceedings of the 7th International Conference Supply on the wings, AIRTEC, Frankfurt.
10. McConnell, S. (1996). “Rapid Development: Taming Wild Software Schedules”. Microsoft Press.
11. Panko, R. R. (1998). “What We Know About Spreadsheet Errors”. Journal of End User Computing’s 10 (2), 15 – 21.
12. Tiwari, S., G. Fadel, P. Koch and K. Deb (2008). “AMGA: An Archive-based Micro Genetic Algorithm for Multi-objective Optimization”. In GECCO’08: Proceedings of the 10th Annual Conference on Genetic and Evolutional Computation.