Deutscher Luft- und Raumfahrtkongress 2013
DocumentID: 301340

PARADISE FOR PRE-DESIGNING AIRCRAFT AND SYSTEMS

H. Schumann, A. Berres, S. Escher, T. Stehr, S. Fricke
German Aerospace Center (DLR), Institute of Flight Systems,
Lilienthalplatz 7, 38108 Braunschweig, Germany

Abstract
A general and pragmatic method for pre-designing technical systems is presented. The method consists of a predesign process and a supporting tool called ParADISE. The method's applicability to different application domains, like aerospace and wind energy, is shown by two examples. In the first example, a wind turbine is designed and in a second one, a short- to mid-range aircraft like the Airbus A320 is designed based on formulas provided by an academic aircraft sizing approach. It is shown, that the method is independent from a specific application domain, which implies that it is applicable to a wide range of technical systems including unconventional configurations.

1. INTRODUCTION

The early identification of risks and costs is a major key for the successful development of any technical system. Up to 85 percent of the total life-cycle costs are determined early during the conception and design stages [1]. To enable a sound decision making in those stages, an early and holistic understanding of the system, together with the complex correlations of its elements, is fundamental.

The amount of information needed to build and operate a civil aircraft, for instance, can easily exceed a 100 meter tall tower of documents. For a single engineer, it is impossible to oversee all relevant dependencies or to draw conclusions within such amount of information.
Additionally, knowledge from several engineering disciplines is needed to precisely understand, for example, the complex consequences of taking bleed air from a jet engine to allow an active flow control on flaps.

Multidisciplinary design teams are established to master the complexity and information need mentioned above. Those teams use design tools to forecast and analyze the system's behavior and they apply design models to highlight different aspects of the system while ensuring a basic common understanding among all engineering disciplines.

In the field of aeronautics, multidisciplinary design has been established for several decades. The evolution led to today's typical aircraft configurations. Because most of current design processes, tools, and design models has been specialized for (and validated by) those classical configurations, the means for analyzing unconventional aircraft are limited. This issue becomes more problematic by the statement of an aircraft manufacturer who stated that the potential for further significant optimization of conventional configurations seems to be more and more limited also.

Contrariwise, the widely agreed and challenging goals of the *Advisory Council for Aeronautics Research in Europe* (ACARE) related to e.g. sustainability require bigger steps in optimization [2]. Those steps may only be achieved by a so-called "game changer", an innovative and unconventional aircraft configuration. But as indicated before, most of available design processes and tools are not flexible enough to handle those unconventional configurations.

This paper presents: 1) a general predesign process for technical systems, 2) DLR's flexible predesign tool *ParADISE* supporting the process together with the tool's modeling, calculation, and export capabilities, and 3) one example from the wind energy domain and one example from the aeronautics domain, which implies the tool's applicability to a wide range of systems including unconventional configurations.

2. GENERAL PREDESIGN PROCESS

The general predesign process consists of a problem analysis and a solution synthesis part. Each part is based on a best practice. The Functions-Based Systems Engineering Method [1] provides a pragmatic way to handle the problem or requirements analysis. The Architectural Design Process defined by ISO 15288 [3] provides a commonly agreed top-level description of a design process for systems in general.

2.1. Problem Analysis

Before thinking of any design solution, the problem must be clearly understood in terms of required system functionality. The Functions-Based Systems Engineering Method, also known as Function(al)-Driven Design, helps identifying the required system functions and supports the separation of thinking about the problem from thinking about solutions by focusing on what the system shall do, not on how it will do. It iterates with the process for analyzing system requirements. The method further "provides foundation for defining the system architecture through the allocation of functions and sub-functions to hardware/software, facilities and operations" [1]. Every component, which may satisfy multiple system functions, may provide general savings for weight, cost, and operation.

After identification and decomposition of system functions, those functions may be further specified by quantitative requirements in terms of expected value ranges of key performance parameters. The transport capacity may be an example for such parameter. The expected value ranges are needed for validation. Notice that some functions or requirements are valid only for a specific operating mode or scenario. For this case, the mentioned Functions-Based Method is enhanced by connecting a state machine in the way that the required values are assigned to those states they are valid for.
In this way, defining requirements becomes more pragmatic which reduces specification time.

2.2. Solution Synthesis

The purpose of the Architectural Design Process is, as stated in ISO 15288, "to synthesize a solution that satisfies system requirements" [3] as well as system functions. The process reflects a top-down approach which starts with the identification and hierarchical decomposition of logical components satisfying one or more of the required system functions. After all functions are allocated to a realizing component via trace links, the logical components are further refined by deciding which parts of them are to be implemented by physical hardware or software. After that, the resulting physical components and its interfaces are specified in a way that allows their full implementation.

In parallel of defining the structure of the system, as described above, its behavior must be also considered. Because it is unfeasible to closely specify a continuous time dynamic behavior within a short-time activity like predesign, the authors prefer to draft the system's behavior in a discrete form by a state machine. Such state machine is able to represent different mission scenarios, operating modes, configurations, flight phases, failure cases, and so on.

After that, key performance parameters are defined for every component of the system-of-interest. The related parameter values are estimated or calculated based on general physical formulas. This allows a first "sizing" of the system-of-interest. Calculated parameters, e.g. the total system's weight, can be validated against expected value ranges identified by the problem analysis. Some parameter values may be changed just to examine their dependencies or impact on other parameters (sensitivity analysis). In addition, first trade-off studies can be performed to rank different solution proposals for being worth further optimization and detailed design, which reduces the time needed for predesign and optimization cycles.

2.3. Process Outputs

The task order of the predesign process and the resulting outputs are illustrated in Figure 1. The numbers on the left side represent the tasks order and on the right side, there is the output of the corresponding task depicted. Tasks 1 and 2 are related to the problem analysis. They create the hierarchical decomposition of system functions and a list of required parameter value ranges. Notice that for each defined scenario or system state, an own parameter list may be specified.

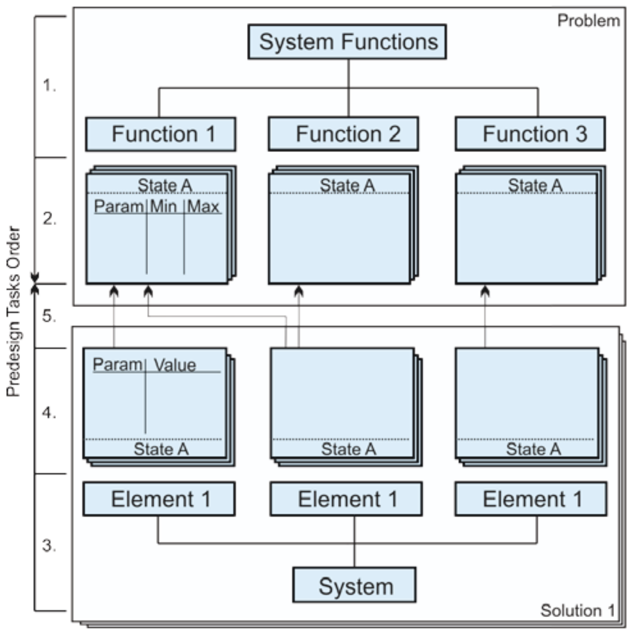
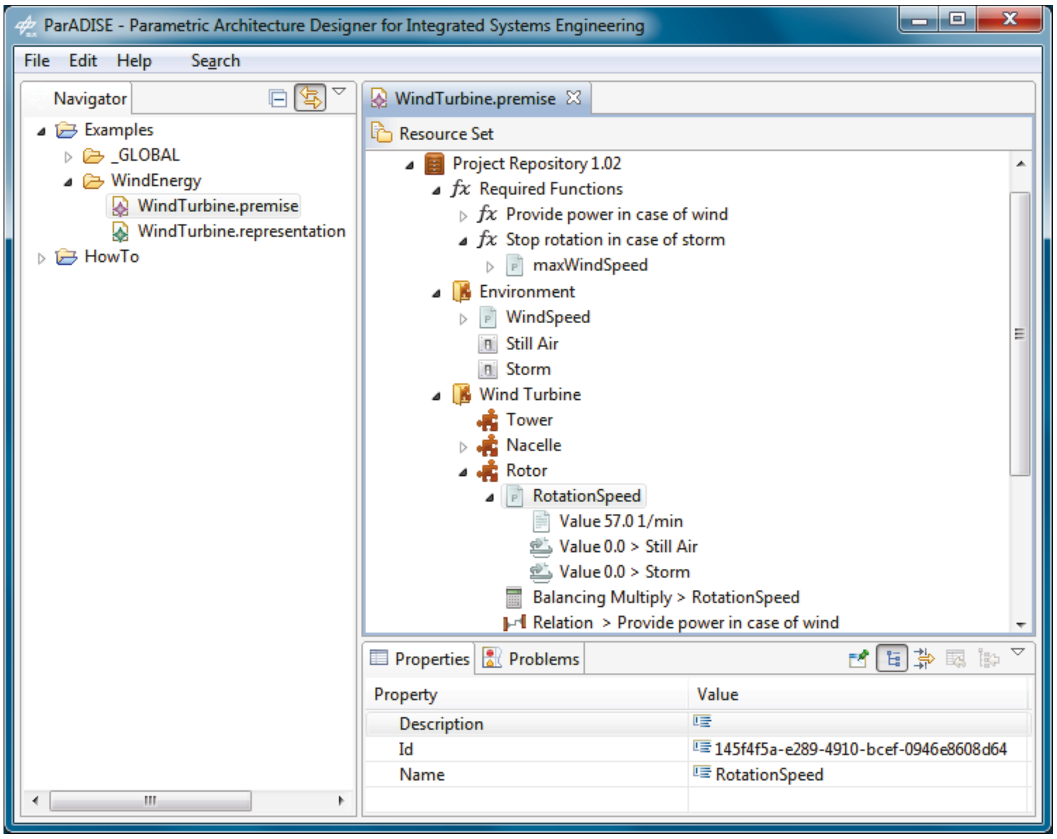


Figure 1: Outputs of the predesign process tasks

Tasks 3 and 4 are related to the solution synthesis. They create the hierarchical decomposition of system elements and a list of calculated parameter values for every system state and every solution considered. In the last task, every element is linked to a function which is, even though partially, satisfied by that element.


ParADISE - Parametric Architecture Designer for Integrated Systems Engineering
File
Edit
Help
Search
Navigator
WindTurbine.premise
Resource Set
Properties
Problems

Examples
_GLOBAL
WindEnergy
WindTurbine.premise
WindTurbine.representation
HowTo

Project Repository 1.02
Required Functions
Provide power in case of wind
Stop rotation in case of storm
maxWindSpeed
Environment
WindSpeed
Still Air
Storm
Wind Turbine
Tower
Nacelle
Rotor
RotationSpeed
Value 57.0 1/min
Value 0.0 > Still Air
Value 0.0 > Storm
Balancing Multiply > RotationSpeed
Relation > Provide power in case of wind

Property	Value
Description	
Id	145f4f5a-e289-4910-bcef-0946e8608d64
Name	RotationSpeed

Figure 2: ParADISE tool with a wind turbine model

3. PREDESIGN TOOL PARADISE

To support the general predesign process in the way that it can be performed for multiple solution proposals in a few days, DLR developed a tool named “Parametric Architecture Designer for Integrated Systems Engineering”, short: *ParADISE*. It provides an efficient and intuitive modeling environment and it is abstract enough to be able to represent a wide range of technical systems. In detail, it supports:

1) Modeling of system architectures
2) Calculation of key performance parameters
3) Generating files for simulation and engineering tools

The user interface of ParADISE is divided into three parts as shown in Figure 2. On the left side, there is a *Navigator* view displaying projects, folders, and model files. In the upper right part, there is the *Model Editor* depicted displaying the contents of the currently opened model files in a tree-like structure. The bottom right of the figure shows the *Properties* view which allows the modification of any property and value related to the currently selected model component.
The model files are stored in the open, XML-based format called *PrEMISE* (Pragmatic Engineering Model for Integrated Systems Engineering) [4] developed by DLR.

Based on the example of a wind turbine, the next subchapters describe how ParADISE can be used to model the architecture, to calculate key parameters, and to export diagrams from the model.

3.1. Wind Turbine Modeling Capability

In the model editor shown in Figure 2, the model file of a wind turbine is opened. In the top of the editor, there is a *Project Repository* component defined which acts as the root container for any component in *PrEMISE* files.

According to the Functions-Based Method mentioned in the predesign process chapter, two required functions for the wind turbine are defined: 1) Provide power in case of wind and 2) Stop rotation in case of storm. Because the definition of what exactly is meant by wind and storm is needed, the functions are each further specified by a quantitative requirement in the form of a parameter like *maxWindSpeed* which expresses a maximum value for the wind speed before the turbine has to stop. Notice that the corresponding value and other parameters are collapsed in the figure and therefore not depicted.

In the next section, an *Environment* container is modeled which provides the *WindSpeed* parameter for different wind scenarios. Besides the default state

Wind, the states *Still Air* and *Storm* are given. The state machine allows to define the transitions between the states and to specify different state-dependent values for the *WindSpeed* parameter.

The last part describes the solution proposal and the related container is called *Wind Turbine*. The turbine consists of the main physical components *Tower*, *Nacelle*, and *Rotor*. The *Nacelle*, for example, consists of further components like a generator and a brake which are collapsed in the figure. The model element shown in the bottom of the editor view, called *Relation*, expresses that this component is allocated to one of the functions, which means that the rotor is satisfying the provide power function, amongst others.

3.2. Calculation Capability

The *Rotor* component is further described by the key performance parameter *RotationSpeed*. The value of this parameter is calculated by a model element called *Balancing*. When all input parameters and a calculation function (like multiplication) are given, the ParADISE tool calculates the rotor speed automatically. In this case, the wind speed is multiplied with a given generator load factor and a brake state value. Because input parameters like wind speed and brake state are dependent from a wind state, different rotation speed values for every wind scenario are generated by ParADISE as shown in the figure.
Because in this case Excel® is used as calculation engine, all calculations can be reviewed and modified by the user, which provides the necessary transparency.

3.3. Data Exchange Capability

ParADISE allows to import and export data from several other textual file formats. For example, a diagram of the system architecture can be exported via the GraphML [5] or SysML [6] format. The related generator provides options to specify which kind of dependencies are to be exported and which not.

Figure 3 shows the generated diagram of the wind turbine dependencies opened with the free graph editor tool *yEd* [7] which also provides the layout generation. Related functions and components are depicted as boxes and the dependencies itself are represented by lines and arrows. The colors for boxes and lines have been separately defined by an additional representation file. It is shown, for example, that the rotor component is dependent from three input parameters as mentioned in the previous subchapter. Additionally, it can be seen that all functions are satisfied by the solution proposal.

In addition, ParADISE offers a generator for Excel to export traceability matrices or to fill custom trade-off sheets with calculated parameter values. Custom MATLAB® scripts can be integrated into the calculation also. A fault tree generator [4] exports failure states and transitions into a related diagram used for fault tree analysis. Another set of generators uses the state machine models to create input files for several Petri net tools. The generator for the multi-body simulation tool SimMechanics™ was already published in [8].

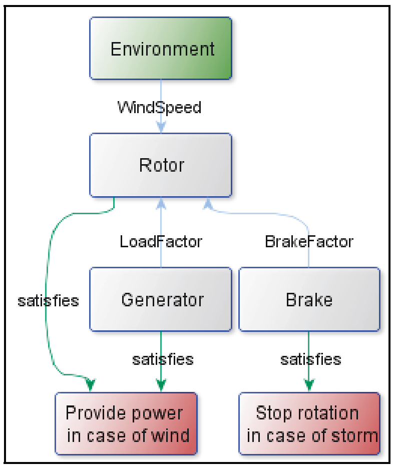


Figure 3: Generated diagram of dependencies

3.4. Concluding Statement

The Function-Driven Design method is supported by ParADISE because of the tool’s capability to express and decompose system functions and related required value ranges (like the minimum wind speed). Different scenarios or states (like still air or storm) can be considered additionally, which shows that the tool supports the entire problem analysis part of the presented predesign process.

The solution synthesis part is supported by the tool because of its capability to express and decompose system elements and to calculate parameter values. Those values can be computed for every specified wind scenario or state. The data exchange capabilities enable the engineer to take over the modeled solution into his specific engineering tool.

Creating the shown wind turbine example was done within 2 hours, which further proves the tool’s applicability for the predesign phase which does not allow spending much time for modeling. An automated validation feature, which compares calculated parameter values with their corresponding expected value ranges, was identified as possible improvement and is planned for one of the next releases.
In the next chapter, a more extensive use case from the aeronautical domain is presented to show the applicability of ParADISE in this field.

4. AERONAUTICAL USE CASE

To create a comprehensible use case from the aeronautical field, a clear and well-described predesign approach for an aircraft is needed. A good base for such kind of process is the early conceptual design process [9], mainly well-described in the academic field.

As predesign example, a short- to mid-range aircraft like the wide-spread Airbus A320 was chosen.

4.1. Conceptual Design Process

A clear process to find a feasible aircraft configuration for given top-level aircraft requirements (TLARs) is described in [10]. The TLARs are therein defined as:

- Required payload, range, and cruise speed
- Landing and take-off field lengths
- Minimum climb gradients for 2nd take-off segment and after a missed approach

The conceptual aircraft design process is supported by the Preliminary Sizing Tool, shortly called *PreSTo* [11], developed by the Hamburg University of Applied Sciences (HAW). PreSTo consists of several Excel® sheets. It is mainly used for educational purposes and assists the user by conceiving a new aircraft design instead of analyzing existing information. In opposite to ParADISE, PreSTo is aircraft-specific.

Based on the given TLARs, PreSTo generally allows the calculation of estimated key parameter values like maximum take-off and landing mass, wing area, wing loads, and thrust-to-weight ratios for different flight phases. The maximum wing load and the minimum thrust-to-weight ratios, which are both limiting the design, are depicted in a related matching chart as illustrated in Figure 4. The chart shown is exemplary based on default values. From such chart, an optimal design point can be deduced as represented by the thick cross in the figure.

To show that ParADISE is also capable of supporting the described aircraft sizing approach, all calculations done by PreSTo have to be integrated into ParADISE. Because PreSTo is based on Excel and ParADISE uses Excel as calculation engine, integrating the PreSTo calculations can be easily achieved as explained in the following section.

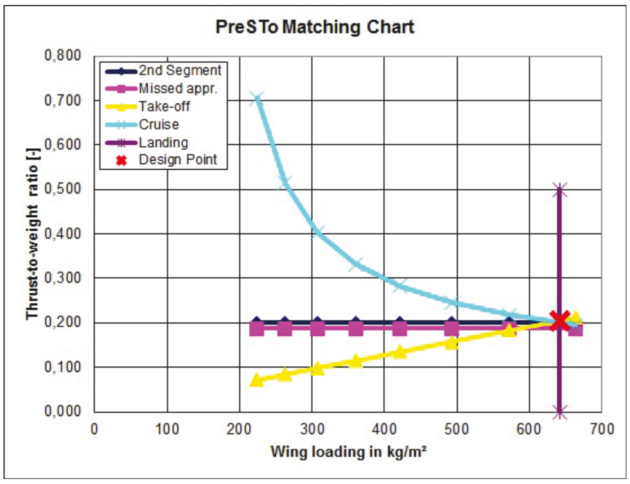


Figure 4: PreSTo chart of design constraints

4.2. Preparation of ParADISE for Calculation

Generally in ParADISE, calculations are defined by a model element called *Balancing*. The user has to specify the parts of the Balancing like source and target parameters and the name of a function which has to be used for calculation. In a mapping file called *Function Pool*, the function names are mapped to a function implementation, in this case, an Excel cell.

For all PreSTo calculations, the Excel cell of the related formula was registered in an own function pool file. This file enables ParADISE to reuse the calculations provided by PreSTo.

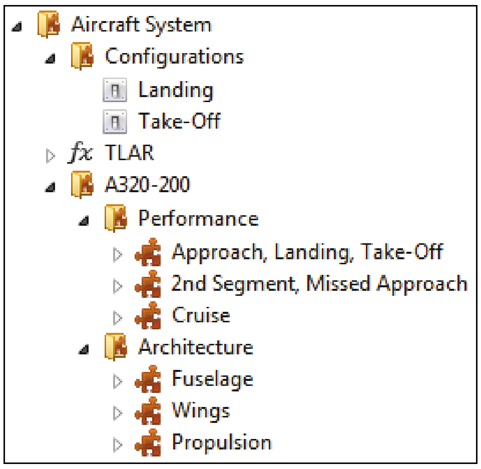
4.3. Predesign of an A320-sized Aircraft

In the following, it is analyzed to which extend ParADISE is capable of supporting the PreSTo predesign approach. Notice that the accurateness of the given and calculated values in the following sections is not in the main focus of this paper. But a realistic example was chosen to make it easier to follow along.

When using PreSTo, different input parameters have to be given and based on them, further performance parameters are calculated. The parameters are assigned to different sections within three Excel sheets. The first section, for example, is called *Approach, Landing, and Take-Off*.

According to that, corresponding sections containing related input parameters, calculations, and calculated parameters were defined in ParADISE. In preparation for modeling an A320-sized aircraft, some top-level aircraft requirements (TLARs) related to an A320 were defined and two special modes were specified, one for landing and one for the take-off configuration. The clear hierarchical top-level structure of the model can be seen in the following

Figure 5.



```
Aircraft System
  Configurations
    Landing
    Take-Off
  TLAR
  A320-200
    Performance
      Approach, Landing, Take-Off
      2nd Segment, Missed Approach
      Cruise
    Architecture
      Fuselage
      Wings
      Propulsion
```

Figure 5: Top-Level Structure of the Aircraft Model

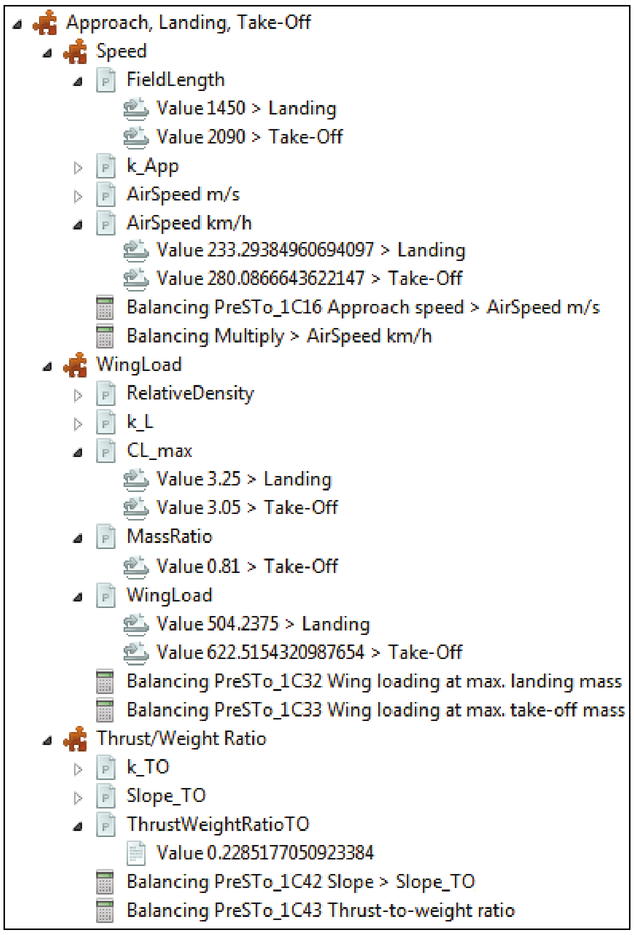
In the following subchapters, the three performance sections and the architecture section are further described.

4.3.1. Approach, Landing, and Take-Off

In the first PreSTo sheet, the landing and take-off field lengths have to be inserted. For the landing as well as the take-off configuration, the maximum lift coefficient has to be specified and the ratio between landing and take-off masses has to be estimated.

In ParADISE, corresponding parameters were defined. Figure 6 shows a model excerpt with the related values, the value units are displayed in a property view (not depicted).
The values for the field lengths (Figure 6) were estimated based on different internet resources. The maximum lift coefficients (*CL_max* in Figure 6) were taken from [12]. The following parameters, also shown in Figure 6, are calculated by ParADISE and correlate with PreSTo. They are reasonable for the real A320:

Approach speed	= 233 km/h
Max. wing load (take-off)	= 623 kg/m²
Thrust-weight ratio (take-off)	= 0,229



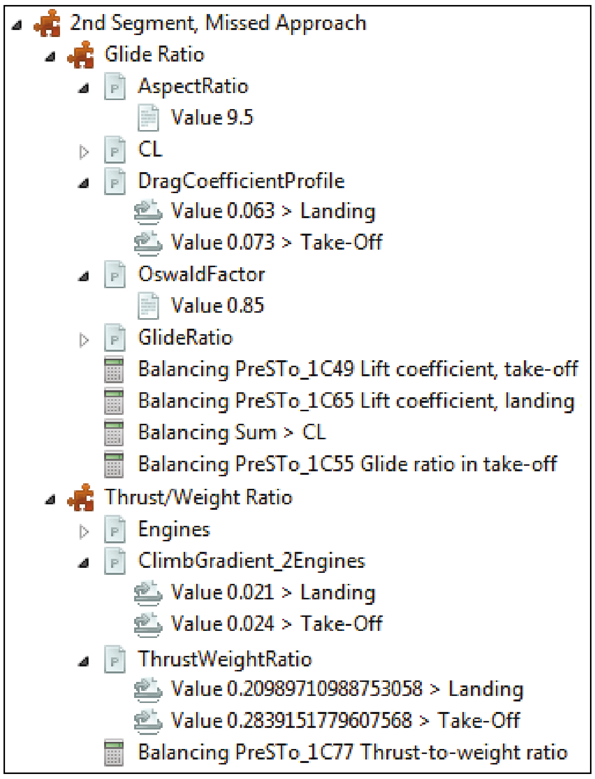
```
Approach, Landing, Take-Off
  Speed
    FieldLength
      Value 1450 > Landing
      Value 2090 > Take-Off
    k_App
    AirSpeed m/s
    AirSpeed km/h
      Value 233.2938496069 4097 > Landing
      Value 280.0866643622147 > Take-Off
    Balancing PreSTo_1C16 Approach speed > AirSpeed m/s
    Balancing Multiply > AirSpeed km/h
  WingLoad
    RelativeDensity
    k_L
    CL_max
      Value 3.25 > Landing
      Value 3.05 > Take-Off
    MassRatio
      Value 0.81 > Take-Off
    WingLoad
      Value 504.2375 > Landing
      Value 622.5154320987654 > Take-Off
    Balancing PreSTo_1C32 Wing loading at max. landing mass
    Balancing PreSTo_1C33 Wing loading at max. take-off mass
  Thrust/Weight Ratio
    k_TO
    Slope_TO
    ThrustWeightRatioTO
      Value 0.2285177050923384
    Balancing PreSTo_1C42 Slope > Slope_TO
    Balancing PreSTo_1C43 Thrust-to-weight ratio
```

Figure 6: Approach, Landing, and Take-Off Performances

4.3.2. Second Segment and Missed Approach

The given parameters are aspect ratio, lift independent drag for cruise mode, and the Oswald efficiency factor, all taken from [12]. Figure 7 shows the model excerpt with the corresponding values. The calculation in PreSTo is based on the certification basis CS-25 [13]. The main resulting parameters were adequately calculated by both tools. They are, as shown in the bottom of Figure 7:

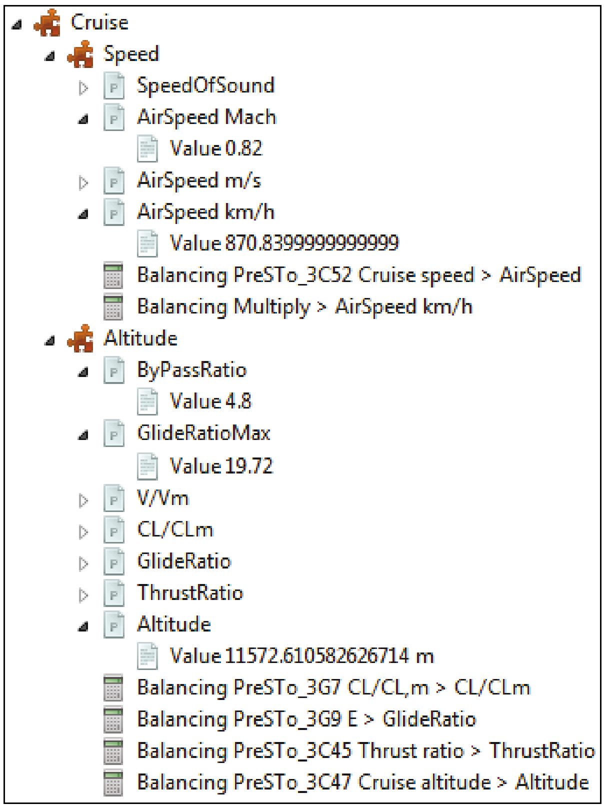
Thrust-weight ratio (2nd seg.)	= 0,284
Thrust-weight ratio (missed)	= 0,210



```
2nd Segment, Missed Approach
  Glide Ratio
    AspectRatio
      Value 9.5
    CL
    DragCoefficientProfile
      Value 0.063 > Landing
      Value 0.073 > Take-Off
    OswaldFactor
      Value 0.85
    GlideRatio
    Balancing PreSTo_1C49 Lift coefficient, take-off
    Balancing PreSTo_1C65 Lift coefficient, landing
    Balancing Sum > CL
    Balancing PreSTo_1C55 Glide ratio in take-off
  Thrust/Weight Ratio
    Engines
    ClimbGradient_2Engines
      Value 0.021 > Landing
      Value 0.024 > Take-Off
    ThrustWeightRatio
      Value 0.20989710988753058 > Landing
      Value 0.2839151779607568 > Take-Off
    Balancing PreSTo_1C77 Thrust-to-weight ratio
```

Figure 7: 2nd Segment and Missed Approach Performances

4.3.3. Cruise



```
Cruise
  Speed
    SpeedOfSound
    AirSpeed Mach
      Value 0.82
    AirSpeed m/s
    AirSpeed km/h
      Value 870.8399999999999
    Balancing PreSTo_3C52 Cruise speed > AirSpeed
    Balancing Multiply > AirSpeed km/h
  Altitude
    ByPassRatio
      Value 4.8
    GlideRatioMax
      Value 19.72
    V/Vm
    CL/CLm
    GlideRatio
    ThrustRatio
    Altitude
      Value 11572.610582626714 m
    Balancing PreSTo_3G7 CL/CL,m > CL/CLm
    Balancing PreSTo_3G9 E > GlideRatio
    Balancing PreSTo_3C45 Thrust ratio > ThrustRatio
    Balancing PreSTo_3C47 Cruise altitude > Altitude
```

Figure 8: Cruise Performances

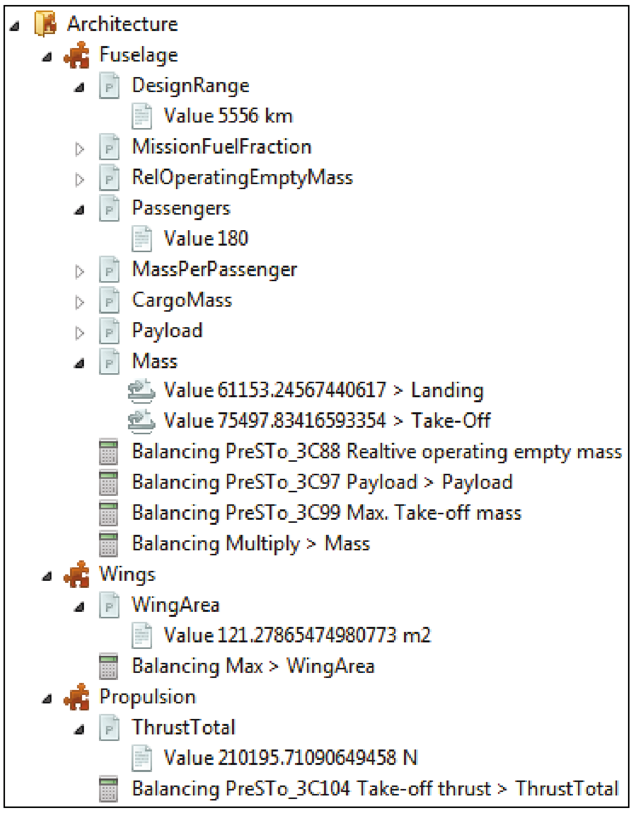
For the cruise mode, Figure 8 shows again the model excerpt with the corresponding values. The cruise speed (*Airspeed* in Figure 8) and the engine's by-pass ratio were given and their related values, as shown in Figure 8, were estimated based on different internet resources. The following parameters were adequately calculated by both tools:

Max. glide ratio	= 19,72
Cruise altitude	= 11573 m

4.3.4. Key Parameters of Architecture

For the architecture section of the model, Figure 9 shows the model excerpt with the corresponding values. The design range and the number of passengers were given based on different internet resources. The following parameters, as also shown in Figure 9, were adequately calculated by both tools and they are reasonable for the A320:

Max. take-off mass	= 75498 kg
Wing area	= 121 m²
Take-off thrust (both engines)	= 210196 N



```
Architecture
  Fuselage
    DesignRange
      Value 5556 km
    MissionFuelFraction
    RelOperatingEmptyMass
    Passengers
      Value 180
    MassPerPassenger
    CargoMass
    Payload
    Mass
      Value 61153.24567440617 > Landing
      Value 75497.83416593354 > Take-Off
    Balancing PreSTo_3C88 Realtive operating empty mass
    Balancing PreSTo_3C97 Payload > Payload
    Balancing PreSTo_3C99 Max. Take-off mass
    Balancing Multiply > Mass
  Wings
    WingArea
      Value 121.27865474980773 m2
    Balancing Max > WingArea
  Propulsion
    ThrustTotal
      Value 210195.71090649458 N
    Balancing PreSTo_3C104 Take-off thrust > ThrustTotal
```

Figure 9: Architecture with Key Parameters

As mentioned before, the accurateness of values is not in the focus of this paper. Nevertheless, the values are all within a 10% margin of published A320 data, which should be well enough for the predesign phase.

5. CONCLUSIONS

In Europe, there is an agreement on achieving challenging goals regarding sustainability in air transportation. Those goals require new and unconventional aircraft configurations. Most of the available predesign methods and tools are based on conventional aircraft with long development life-cycles.

A general predesign process for the early predesign of new technical systems was introduced. The process is based on the ISO 15288 standard and on the *Function-Driven Design* method, both known from on the field of *Systems Engineering*.
The predesign process allows a pragmatic way of defining requirements in terms of system functions. Thinking of the problem is clearly separated from thinking of a solution and different scenarios or system states can be explicitly considered at the same time. The traceability between system elements and system functions is ensured and different solutions can be compared to each other in a convenient way.

A tool called *ParADISE* which supports the predesign process was presented. The tool's applicability to different application domains, like aerospace and wind energy, was shown by two examples.
In the first example, a wind turbine was designed. In a second one, a short- to mid-range aircraft like the Airbus A320 was designed following the aircraft sizing approach of *PreSTo*, an academic sizing and optimization tool for aircrafts. Several key performance parameters were calculated in that example. Because the Excel® sheets of PreSTo were reused, ParADISE calculated the same reasonable results for an aircraft like the A320.

In summary, the benefits of using ParADISE for predesign are:

1) Independence of application domains, which implies that it is applicable to a wide range of technical systems including unconventional configurations
2) A clearly structured top-level overview of models (see Figure 5) and a more graphical user interface than provided by Excel
3) Generation of diagrams visualizing dependencies between system elements, traceability links between system elements and functions (see Figure 3), or c) Fault Trees
4) Further data exchange capabilities for engineering tools like MATLAB® or simulation tools which reduces the time needed for design and optimization cycles
5) The Tool is based on an open, clearly structured XML format with concepts taken from the Systems Engineering community

The comparison of expected values with calculated ones was identified as a useful feature and is planned for one of the next releases of ParADISE, the Parametric Architecture Designer for Integrated Systems Engineering.

REFERENCES

[1] Haskins, C. (Ed.): Systems Engineering Handbook ver. 3.2.2. Technical Product INCOSE-TP-2003-002-03.2.2, International Council on Systems Engineering (INCOSE), 2011

[2] Muller, R.: AGAPE Project Final Report: Publishable Summary. Brussels, 2010. URL: http://ec.europa.eu/research/evaluations/pdf/archive/other_reports_studies_and_documents/agape_final_report_summary.pdf (2012-04-16)

[3] ISO/IEC 15288:2008: Systems and software engineering - System life cycle processes. ISO, 2008

[4] Schumann, H., Berres, A.: Modell-basierter Systementwurf: Generierung von Fault-Trees. German Aerospace Congress, 2012

[5] GraphML Project Group: The GraphML File Format. URL: http://graphml.graphdrawing.org/ (2013-07-18)

[6] Open Management Group (OMG): OMG Systems Modeling Language (SysML). URL: http://www.omgsysml.org/ (2013-07-18)

[7] yWorks GmbH: yEd Graph Editor. URL: http://www.yworks.com/de/products_yed_about.html (2013-07-18)

[8] Schumann, H., Zamov, P., Escher, S.: Model-Based Design and Tool Data Exchange in Aerospace: A Case Study. CEAS Aeronautical Journal, vol. 2 (no. 1-4), pp 295-303, 2011

[9] Torenbeek, E.: Synthesis of Subsonic Airplane Design. Delft University Press, 1982

[10] Abulawi, J., Seeckt, K., Pommers, M., Scholz, D.: Automatic Generation of 3D-CAD Models to Bridge the Gap between Aircraft Preliminary Sizing and Geometric Design. In Proceedings: German Aerospace Congress 2011, Bremen, pp 405-415. ISBN: 978-3-932182-74-X

[11] Seeckt, K., Scholz, D.: Application of the Aircraft Preliminary Sizing Tool PreSTo to Kerosene and Liquid Hydrogen Fueled Regional Freighter Aircraft. In Proceedings: German Aerospace Congress 2010, Hamburg. ISBN: 978-3-932182-67-7

[12] De Gernon, F., La Vecchia, M., Rigaldo, A.: Performance and Design of the Airbus A320: Analysis of a Subsonic Aircraft. 2009. URL: http://perso.centrale-marseille.fr/~arigaldo/visible/KTH_projects/Performances_A320.pdf (2013-07-18)

[13] EASA CS-25: Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes, Amdt. 12. European Aviation Safety Agency (EASA), 2012