Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Project

Department Fahrzeugtechnik und Flugzeugbau

Structure and function of the aircraft design program PrADO

Lionel Salavin

Supervisor: Kolja Seeckt
Examiner: Prof. Dr.-Ing. Dieter Scholz, MSME

Submission date : 19 May 2008

Hochschule für Angewandte Wissenschaften Hamburg
Department Fahrzeugtechnik + Flugzeugbau
Berliner Tor 9
20099 Hamburg

Author: Lionel Salavin
Submission date: 19 May 2008

1. Examiner: Prof. Dr.-Ing. Dieter Scholz, MSME
2. Examiner: Kolja Seeckt

Supervisor: Kolja Seeckt

Abstract

This report is a description of the aircraft design software *PrADO*. PrADO is the name of an aircraft pre design and optimisation program, which already proved its capacities in many projects. The main goal of this report is about helping to understand the functions and structure of PrADO in order to modify it. Therefore, it provides a description of its main possibilities, as a short manual of the program, based on previous paper aiming to document it. As the reports aim to help to understand the structure of the program, it analyzes the organisation of the different files, which constitute the software. It explains what each kind of file does for the program and where it is located in the windows folders. The first distinction between each kind of files shows that the most important files are the Fortran files, which are organised into modules executable and libraries of functions. Therefore, it is relevant to have an overview of the first modules and a short description of the most important libraries. Since this report is part of a cooperation between the HAW and the IFL, which aim to add a turbofan module in PrADO, it observes in detail every module and each important file of libraries related to the propulsion. Particular attention is given to the computation of nnthe turbofan‘s performance.

The end of this report consists of two parts addressed to two different kinds of readers:

-An Analysis of the methodology needed to understand the program is there for any new comer to PrADO’s source code.

-An open discussion on the program is addressed to the PrADO’s programmer to submit some propositions concerning the coding and the automatic generation of documentation for PrADO.

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

DEPARTMENT FAHRZEUGTECHNIK UND FLUGZEUGBAU

Structure and function of the aircraft design program PrADO

Task description for a project 2 in accordance with test conventions

Background

PrADO (Preliminary aircraft Design and Optimisation Program) is a program of the institute for Aircraft Construction and Lightweight Construction of the Technical University of Brunschwig for the iterative, multidisciplinary design process of airplanes. It consists of a large number of subroutines, which reflect the contributions of the most important fields of activity needed in the design. These sub-programs are arranged according to draft problems or configuration to an overall system and supplemented if necessary by missing analyzers. The program is used at the HAW in the research project of the Green Freighter (http://GF.ProfScholz.de).

Task

The existing model is to be examined and documented for the work within PrADO. The modular structure of the program such as the libraries, the routines and subroutines etc. are to be described as a basis for later changes and implementation of modules on the part of the HAW.

The report was written according to German DIN or international standards on report writing.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3D	3 Dimensional
CC	Combustion chamber
DB	Data Bank
DMS	Data Management System
DOC	Direct Operational Cost
DOS	Disk Operating System
FE	Fluid and Energetic
GUI	Graphic User Interface
HPC	High Pressure Compressor
HPT	High Pressure turbine
HTML	Hypertext Mark-up language
IFL	Institut für Flugzeugbau und Leichtbau
LPC	Low Pressure Compressor
LPT	Low Pressure turbine
MD	Module
PC	Personal Computer
PrADO	Preliminary Aircraft Design and Optimisation program

Terms and Definitions

Software

Computer software is a general term used to describe a collection of computer programs, procedures and documentation that perform some tasks on a computer system. The term includes application software such as word processors which perform productive tasks for users, system software such as operating systems, which interface with hardware to provide the necessary services for application software, and middleware which controls and co-ordinates distributed systems. (**www.Wikipedia.org 2008**)

Documentation

In general terms, **documentation** is any communicable material (such as text, video, audio, etc., or combinations thereof) used to explain some attributes of an object, system or procedure. It is often used to mean engineering documentation or software documentation, which is usually paper books or computer readable files (such as HTML pages) that describe the structure and components, or on the other hand, operation, of a system/product.

A common type of software document frequently written by software engineers in the simulation industry is the SDF (software documentation folder). While developing the software for a simulator, which can range from embedded avionics devices to full motion control systems, the engineer keeps a notebook detailing the development lifecycle of the project. The notebook can contain a requirements section, an interface section detailing the communication interface of the software, a notes section to detail the proof of concept attempts to track what worked or didn't work in solving certain problems, and a testing section to detail how the software will be tested to prove conformance to the requirements of the contract. The end result is a detailed description of how the software is designed, how to build and install the software on the target device, and any known weaknesses in the design of the software. This document will allow future developers and maintainers of the trainer to come up to speed on the software design in as short a time as possible and have a documented reference when modifying code or searching for bugs. (**www.Wikipedia.org 2007**)

Fortran

Fortran (previously **FORTRAN**) is a general-purpose, procedural, imperative programming language that is especially suited to numeric computation and scientific computing. Originally developed by IBM in the 1950s for scientific and engineering applications, Fortran came to dominate this area of programming early on, […].

Fortran (a blend word derived from *The IBM Mathematical* ***For****mula* ***Tran****slating System*) encompasses a lineage of versions, each of which evolved to add extensions to the language while usually retaining compatibility with previous versions. Successive versions have added support for processing of character-based data (FORTRAN 77), array programming, module-based programming and object-based programming (Fortran 90 / 95), and object-oriented and generic programming (Fortran 2003). (**www.Wikipedia.org 2007**)

1. Introduction

1.1. Motivation

Today, most aircrafts are based on a very conservative design structure: a simple tail aft fuselage and simple wings. The jet engines follow the same rules: they mostly use classical kerosene, a fossil fuel about to deplete in the upcoming decades. However, gasoline price has kept increasing for twenty years and the increase is going faster and faster: the price for one barrel of crude oil has passed the symbolic 100$, limit which was unimaginable years ago. With such economic pressure general aviation will surely suffer and the more pessimistic of the ecologists are already predicting the end of general aviation in ten years. In fact, if no new fuel is found and their prediction on fuel reserve turns out to be true, then there might be not enough kerosene to allow a viable aircraft economy. Much researches is currently being carried out on the subject and if synthetic fuels seem to be a good short-term answer, new sources of power are still to be studied. Hydrogen is another solution, implying less green house effect, but it involves a complete aircraft reconfiguration, which necessitates a complex analysis of the whole system. Therefore, such exercise needs complex and powerful tools. Many software are capable of performing single analysis on a particular part but very few give the opportunity to realise a complete aircraft design with every parameter variation considered in the global conception and giving answers on the global possibility of a configuration. PrADO is a powerful tool, which can perform such tasks, and HAW is one of the universities at the head of the research on hydrogen aircraft design with the Green Freighter project. In their cooperation, the HAW is currently adding a turbofan module to PrADO. In order to facilitate the comprehension of the program for other students, this project has been proposed to explain the structure of this complex assembly, so that following students will have to spend less time on understanding the program but more on making the program more complete. In this situation, a description of PrADO's structure for the HAW appears not only as a chance to understand the mechanism of Aeronautic conception but also as a great opportunity to be a small part for the future of aeronautic.

1.2. Aim of the project

Changing a simple parameter in an aircraft can become an extensive and complex project, and so is it when you try to add a new concept in a program supposed to help in such a work.

This report helps the analytical process of the program. It shows its possibilities and its structure both on the side of the folders and on the side of the programs: what and where each

file related to the software is. It is also meant as a help to understand the program for following students who will have to perform the changes in PrADO.
The project is part of a collaboration between the IFL and the HAW where the HAW is currently trying to add a turbofan module in PrADO. This is why the description of the program focuses on the engine part of the program.

1.3. Structure of the Project

The main part of the report describes the structure of PrADO including the module system, the data management system and the turbofan parts.

Chapter 2 describes the program concept and most of its possibilities in order to present the possibilities offered by the PrADO.

Chapter 3 provides a description of the global program in terms of folder organization as well as in terms of file's interactions. It includes a description of the first 14 modules algorithm and of the libraries.

Chapter 4 is about the data management system. It presents the different subroutines and how they work.

Chapter 5 describes more precisely the turbofan part of the software. The modules algorithm, as well as the libraries, the data and the templates related will be analysed here.

Chapter 6 is a description of my analysing methodology. It should help any newcomer to the program to find how to get the file he is looking for or the possibilities behind any button.

Chapter 7 is an open discussion on the possibilities to make more complete an evolving program description based on other documentations examples.

Appendix A Code of the SOURCE/MAIN/MD6.for files

Appendix B Code of the SOURCE/LIB/TA2_LIB/MD6.for

2. Generalities about PrADO

A big part of the possibilities and utilizations of PrADO are described **in KIESEL 2007**: "Methodisches Entwerfen von Verkehrsflugzeugen mit PrADO" and **HEINZE 2001** "Multidisziplinäres Flugzeugentwurfsverfahren PrADOProgrammentwurf
und Anwendung im Rahmen von Flugzeug-Konzeptstudien". This shortened description presents the program in English to any newcomer. It shows guidelines for using it to conceive an aircraft and its basic principles.

2.1 Program Concept

The program PrADO (Preliminary Aircraft Design and Optimisation program) copies the interconnected, iterative draft process of the technical disciplines involved. It consists of a multiplicity of subordinated Fortran programs, which are organised in so-called libraries. In the version available here, there are around 1500 Fortran source files. The libraries represent the actual system core, since they contain the mathematical process and the model of conceptions. The draft program divides itself according to the tasks of the conceptions of airplanes in four levels, which are related with one another all together by a *data management system* (DMS) as shown in Figure 2.1.

The DMS creates 15 files thematically arranged, which contain the design variables (e.g. geometry, aerodynamic factors, masses, etc.) from a "Master Input files" or "specification files", which contain the data needed for the draft. The operator fills it before he uses PrADO. The program reads the input values, improves them in iterative steps and overwrites finally the initial values. Thus, the data always reflect the latest computation. At the end, the data banks created in such a way describe the desired airplane configuration and its perfomances. Each program level has its own functions:

The first of the four levels includes the subroutines, which produces the databases of the DMS out of the specification files of the user. More subroutines serves to display the result of a computation in form of table or graphics (technical diagram, 3D model of aircraft, etc…) . the Graphical User Interface, which is programmed in Java, is also in this level. As more detailed later, the GUI is a java program, which reads a script. Those scripts are written in a very user - friendly syntax and they can be easily modified through the windows notepad. This part is more detailed in chapter 6.

The second level contains different mathematic process for an effective optimisation of the design variables of a specific configuration, which will have other parameter fixed (e.g. the geometrical configuration, the number of engines.). The aim of this part is the maximisation/minimisation of specified variable (e.g; the direct operational cost for a liner) and the verification that the optimised aircraft respect the given limit conditions (e.g; the distance for landing/take-off).

The actual interdisciplinary design process takes place in the third level. The draft divides itself into various subtasks (e.g. *Determination of airplane geometry*, *computation of the aerodynamic characteristics*, etc.). The subdivisions are called modules and they reflect all fields related to the draft. They are executed one by one, and they all computes the so-called dependant design variables (e.g. fuel mass, static thrust, or MTOW) and exchange data only with the database. The program executes them iteratively during the draft process until the dependent design variables converge. From the input values of the user and/or the results of other draft modules, they compute new current values like e.g. the fuel mass or the static thrust.

The fourth and thus last level contains the problem-oriented program libraries and forms thereby the heart of PrADO. These program libraries contain the physical computation models for determination of the variables of the atmospheric, aerodynamic or again flight mechanic domain. Thoses subroutines are called in the design modules of the third level. They contain also a lot of process for the determination of the differents parts mass (e.g. statistic, FE-process for determination of the wing structure’s mass) or also for description of the turbojet properties.

The modular concept of PrADO leads to a high flexibility of the application. Design problems can be quite easily eliminated. In addition, the appropriate modules can be replaced or inserted. Another advantage is that PrADO can call external programs, which receive then from it the desired information via a data transfer. [1]

[1] Subchapter translated from **Heinze?**, p.2&3

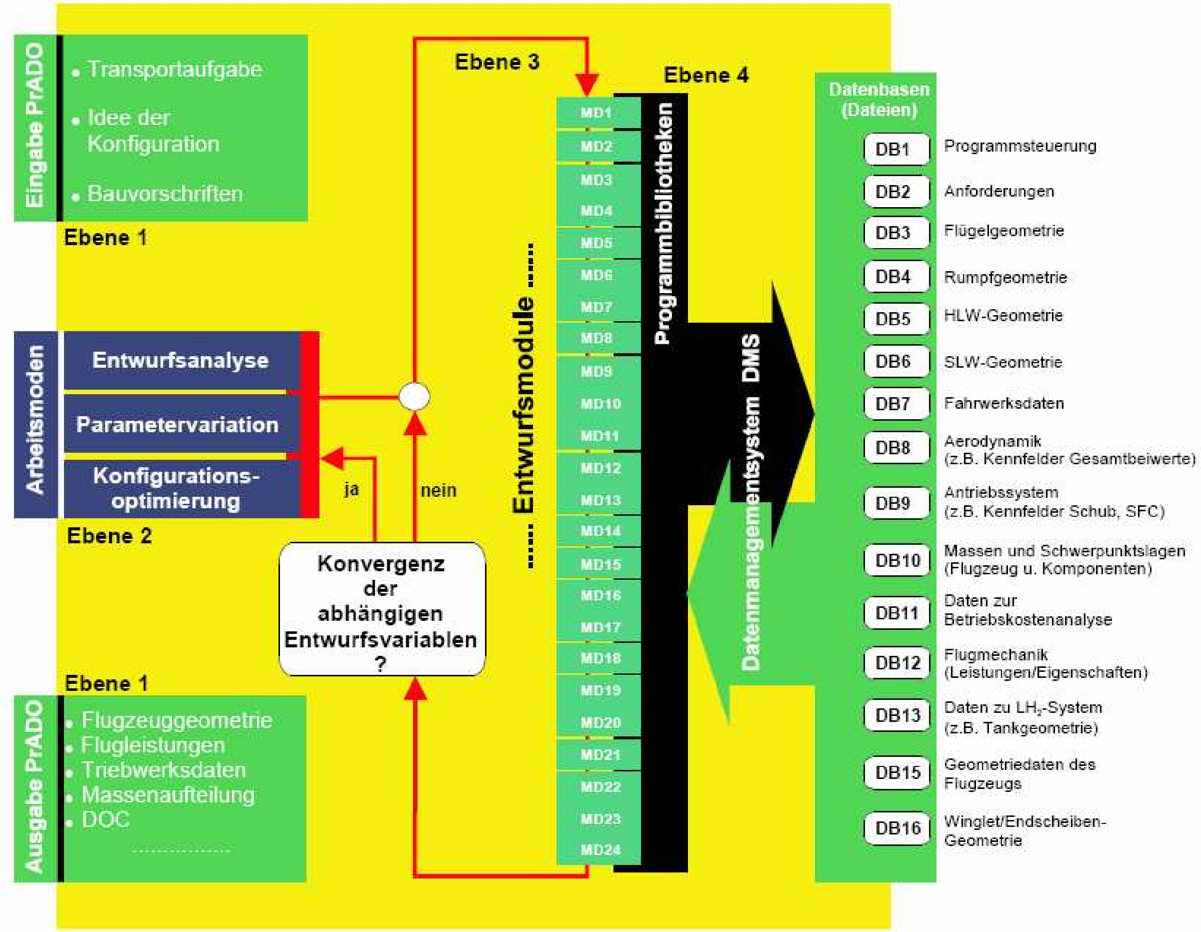

Figure 2.1 schematic program structures (from **IFL**)

2.2 The IFL

The Institut für Flugzeugbau und Leichtbau was created in 1983 in Brunschwig. In the first years, the main point of research was the area of slow-speed flight and short start properties. Above all the development of the slow airplane "zaunkoenig" is to be mentioned here under the direction at that time of Professor H. WINTERS.

After the readmission of building airplanes in Germany, starting from 1955 the research of operational strengths with GRP components materials, in particular with gliders, extended the sphere of activity of the IFL.

Today, the Institute deals with a set of different, but interconnected ranges of topics. On the one hand and since the eighties, it has been carrying out research on the range of airplane's pre development and concepts and on the other hand the development of new lightweight structures. Its concern in the multidisciplinary optimization of aircraft concepts led them to develop PrADO. Many projects used it, not only for the main customer Airbus, but also for many other aviation companies.[2]

2.3 External Programs Needed

As already mentioned PrADO works in interaction with different external programs and as a program itself it is generated by the use of different compiler.
Here follows a list of the different program used with the version at disposal:

- As FORTRAN Compiler, Microsoft Compaq Visual FORTRAN 6 is used
- For the Graphical output, TECPLOT 360 (a CFD & Numerical Simulation Visualization Software) is used
- The calculator and the notepad of the windows accessories list are used at different times by PrADO
- JAVA, from the firm SUN MICROSYSTEMS, is the language of the GUI program.

2.4 Main Functions

The Main Windows of PrADO consists of one menu, three operational windows and a status-monitor as fig. 4.1 shows.
The status-monitor shows every program and/or script launched by the main windows. It is very useful to find out which button calls which subroutine.
The buttons of the middle control panel form the main part of the program control.

[2] Subchapter translated from **Kiesel 2007**, p.14

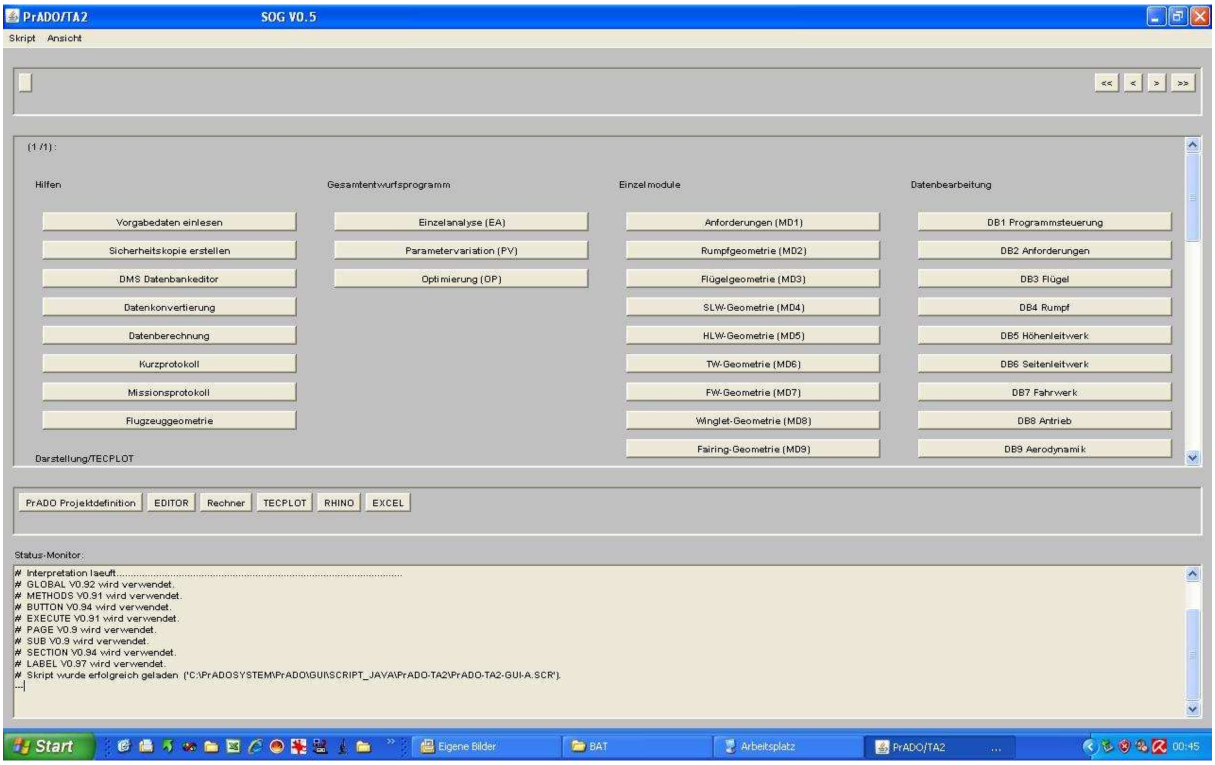

Figure 2.2 Main Window of PrADO (From **Kiesel 2007**)

This chapter explains the possibilities given by this main window.
At first, a project must be defined. If the button „*PrADO-Projektdefinition*“is clicked, then a secondary interface (another java window) opens, as fig. 4.4 shows. There two new possibilities appear for the user. On the one hand, with the button „*Letzte GUI-Einstellungen einlesen*“ the last project worked can be loaded again or a new project can be defined with the second options, „*vordefinierte Verzeichnisse*“ and „*neues Verzeichnis*“.. If a new work is to be started, then the last of the possibilities given (*** NEW *** unten angeben *** NEW ***) must be selected.

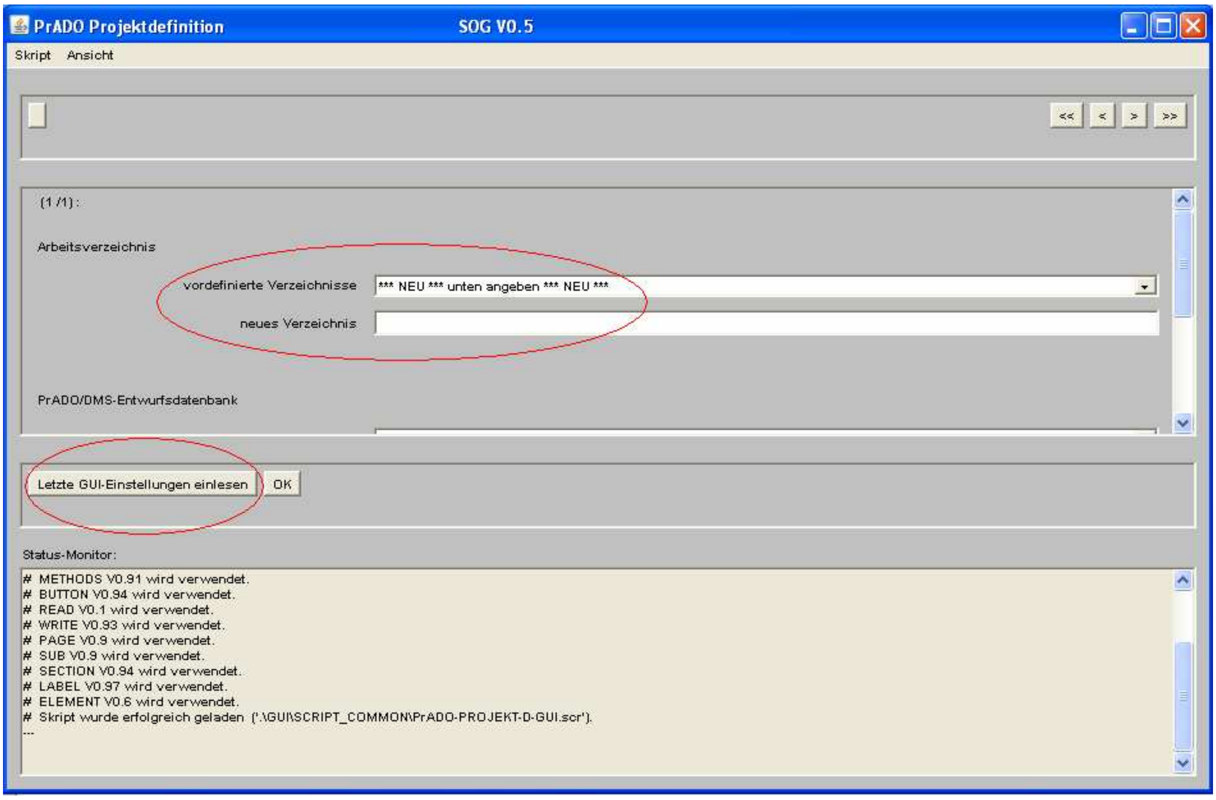

Figure 2.3 Definition of PrADO Project (from **Kiesel 2007**)

In the next step the file "PrADO-TA2" must be opened. There, one looks for the desired project file and copies its address. Finally, the complete headline (everything behind the field address) is selected and the address is passed. The line should look as follows:

- *C:\PrADOSYSTEM\PRADO\PROJEKTE\PrADO-TA2\....(Projektname)*

Using copy and paste, one can now fill the second line of the inputs windows and validate with ok.

Thus, the project is defined and PrADO knows the destination folder. It stores all necessary data, and all project files will now refer to it.[3]

[3] Subchapter translated from **Kiesel 2007,** p.28

2.4.1 Helps Functions

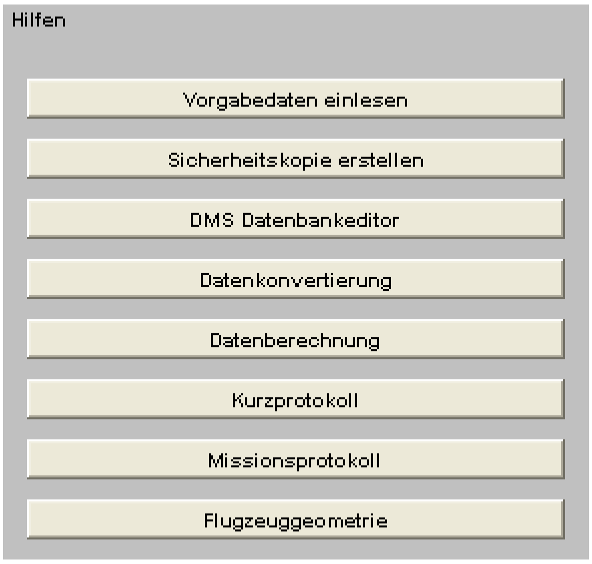

Figure 2.4 Help functions (from **Kiesel 2007**)

Within this range, the user finds eight different options, as figure 4.5 shows
With the first button, the “Master Input file” or specification file can be read in. This button opens a new window permitting to indicate the access to the specification file. It shows also that the latest project address is correct. If this is the case, then the name of the specification file is still in the input windows. The user must repeat this procedure and confirm it with Return.

The second button creates a backup copy. The process principles are the same as for the previous button. A similar window appears with the request to enter the desired file. After successful input of the project file, a new file is created.

With the "DMS data bank editor», all known used variables can be displayed, either per data base or with all data bases altogether. If the user enters e.g. DB1 in the first line and then ticks the field "variable list of the selected data base", then this window will indicate all variables. In addition, a text file with additional explanations is stored in the project file.

The button "data conversion" gives the opportunity to convert data files in order to have them working with external programs such as Tecplot, Nastran, etc...
The button "data computation" opens a new window, which permits specific fuselage variable computations.
With the button “short protocol”, a compact listing of all important draft data can be indicated. Similarly functions with the choice of "mission protocol".

The last button of this border treats the specification file through two iteration steps producing a meaningful solution in a 3D diagram. [4]

2.4.2 General Design Programs

There are three kinds of the total draft programs:

- The single analysis
- The parameter variation
- The optimization

All three presuppose a complete specification file, which does not have to be compellingly optimal however. Their tasks consist of discovering and improving not meaningful inputs. They use most diverse mathematical procedures in order to come to the most effective optimization of free draft variables of the examined configuration. Beyond that, it is the task of these programs to reach a maximization/minimization of the critical variables and examine the boundary conditions. The results are collected in the project file. The individual calculation takes a certain amount of time. For instance, the time needed for a parameter variation is around 14 hours.[5]

[4] Subchapter translated from **Kiesel 2007,** p.28
[5] Subchapter translated from **Kiesel 2007**, p.32

2.4.3 Single Modules

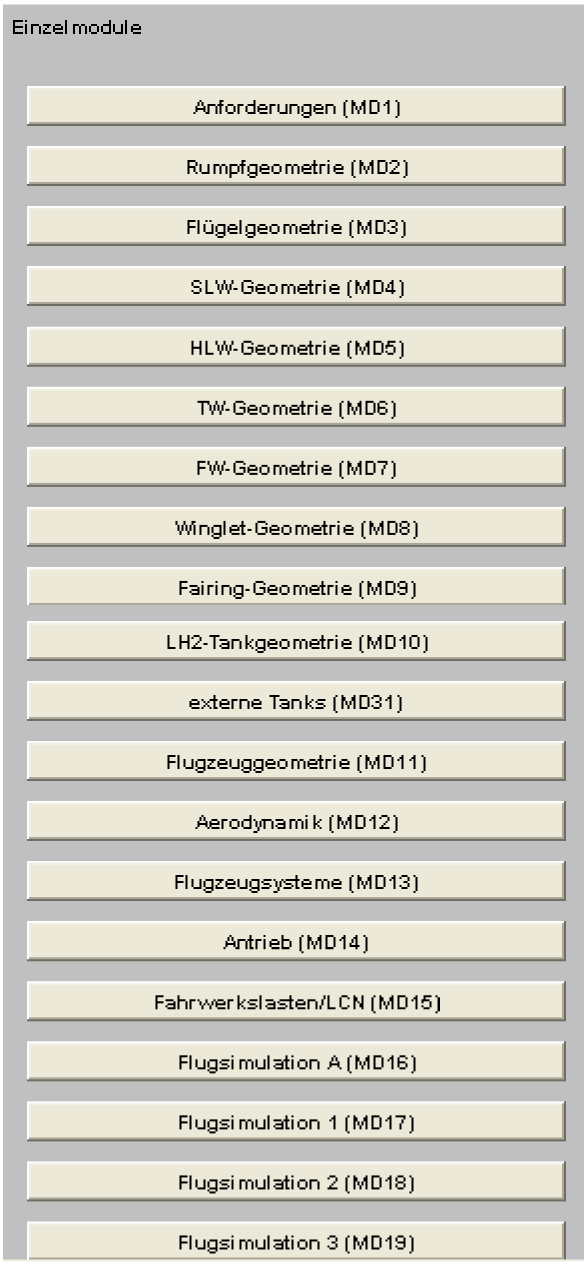

Figure 2.5 List of the modules (from **Kiesel 2007**)

Behind these modules hides itself the actual interdisciplinary design process for the examined configuration.

The draft activity has been arranged into clear subtasks, whereby each one stands for a field of activity involved in the draft. The partitioning was made in so-called draft modules. They compute the current values of the dependent draft sizes from the inputs of the user or the results of other draft modules (e.g. Fuel mass, runway length, max. take-off weight, etc.). The draft iteration is finished if the dependent draft variable shows a convergence. If someone wants to look at geometry of its fuselage once he has started studying, then the complete airplane geometry does not have to be designed, but the program can operate MD1 „requirements “and MD2 „Fuselage geometry “ : only the fuselage will be computed. Then, with the help of the graphical output program, the fuselage alone can be represented.[6]

[6] Subchapter translated from **Kiesel 2007**,p.32

2.4.4 Dealing with Data

By clicking on the data bank buttons on the right of the main windows, all design variables are shown. A new window appears, with more options and with the appropriate outputs. By use of the GUI „reading DB“, the variables sizes are loaded and represented in appropriate places. Therefore, by clicking on each data bank, it is possible to examine all design data. There are some data banks which provide a link to the module they are related to. They can be executed exactly like on the Main Window.

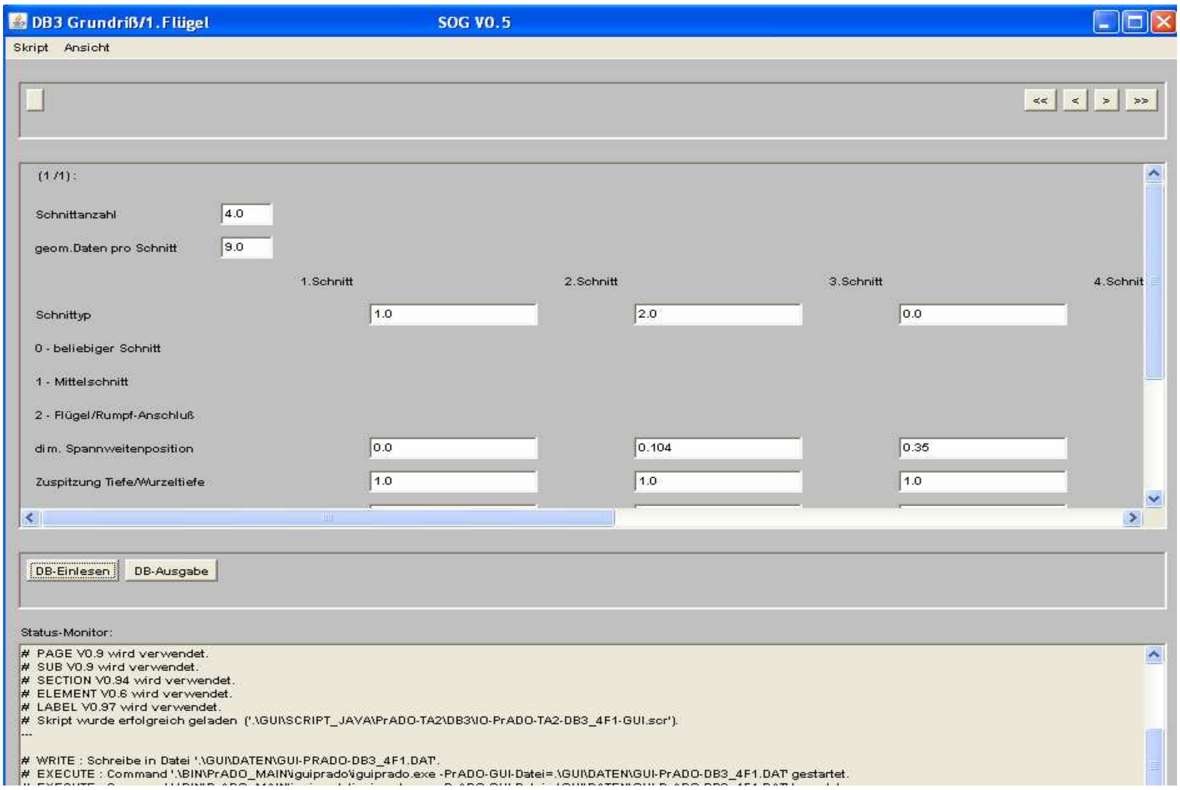

Figure 2.6 GUI of the third data bank (from **Kiesel 2007**)

2.5 Graphics/TECPLOT

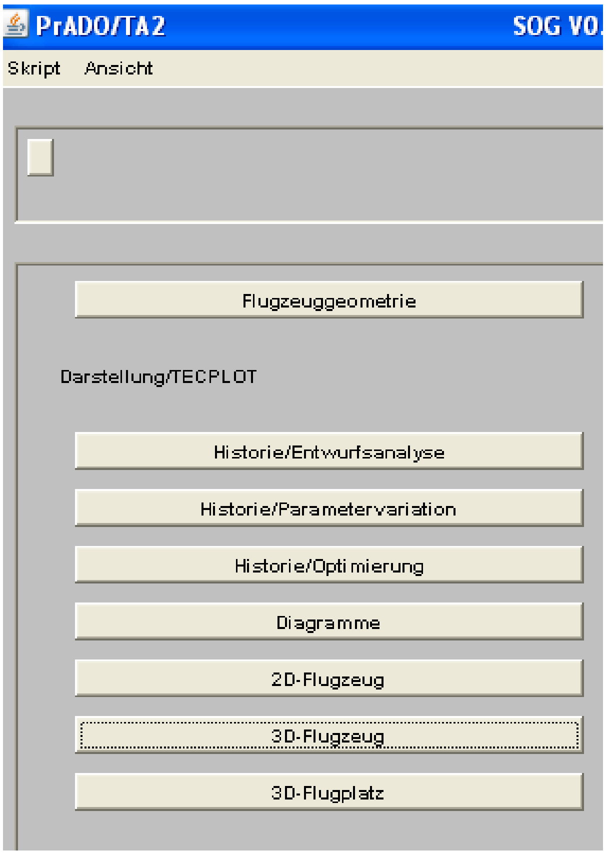

Figure 2.7 Possibilities with TECPLOT (from **Kiesel 2007**)

The pictures below show the possibilities of graphical output with PrADO.
The three first buttons open new GUIs, which can represent a history of the individual total draft programs. The result takes place in a diagram produced by TECPLOT.

A complete airplane configuration, alternatively as 2D or 3D Model, can be represented with the two following buttons. The data given by the graphics depends on the quantity of the modules which where run before by the program. If one launches for example only MD1 (*requirements*) and MD2 (*fuselage geometry*), then only the fuselage can be displayed as graphic. [7]

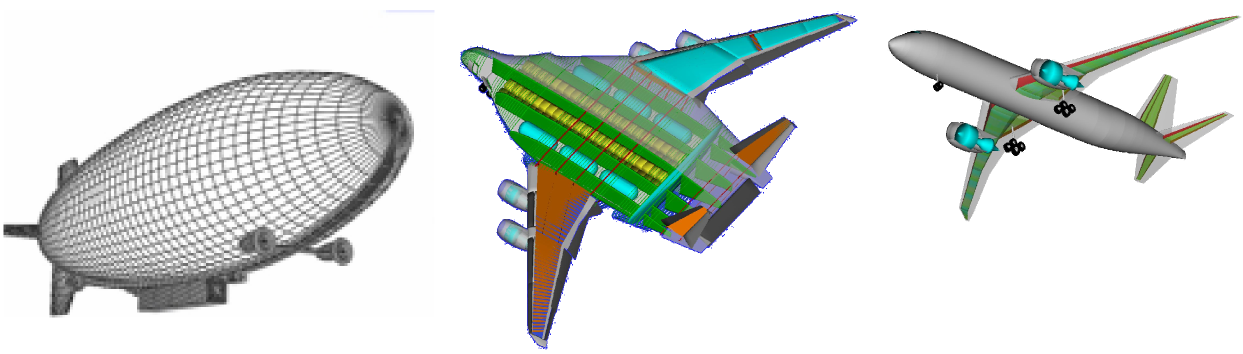

Figure 2.8 Different 3D model generated with PrADO (from **IFL**)

[7] Subchapter translated from **Kiesel 2007**, p.35

3. Description of the global Structure

3.1. Description of the global organisation

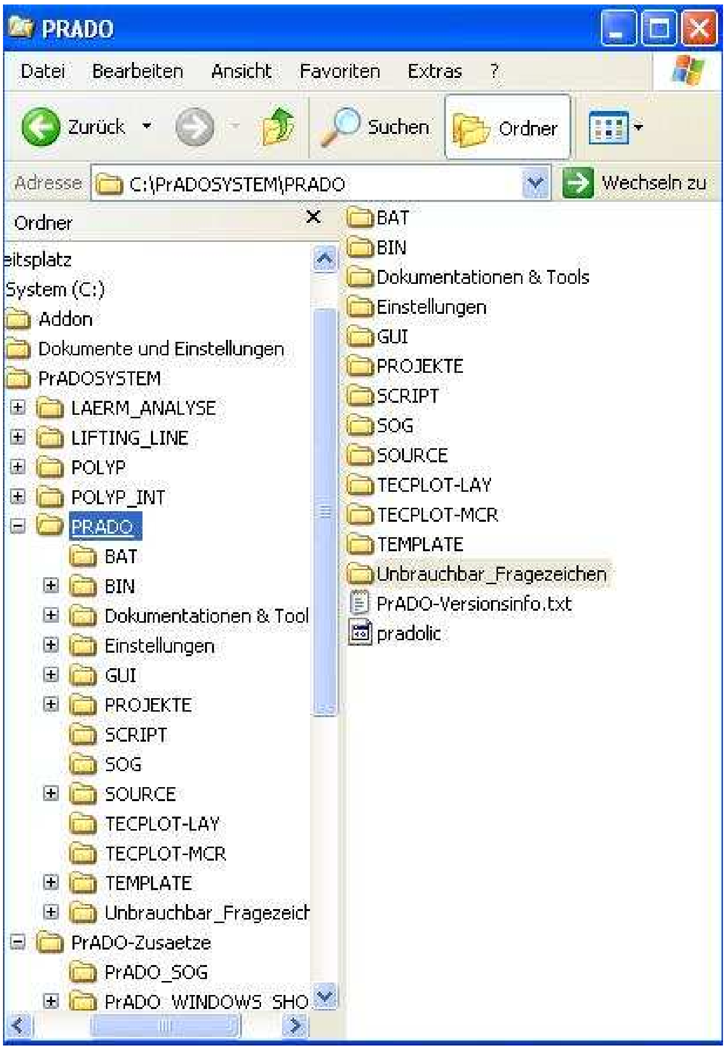

Figure 3.1 Content of the PrADO Folder

The main PrADO folder is in

C:\PrADOSYSTEM\PRADO :

It contains most of the folders needed to have the principal PrADO functionality working:

As it can be deduced from this screenshot, each folder described in this chapter is directly stored in this folder.

The following schema shows globally what are the more important folders and files:

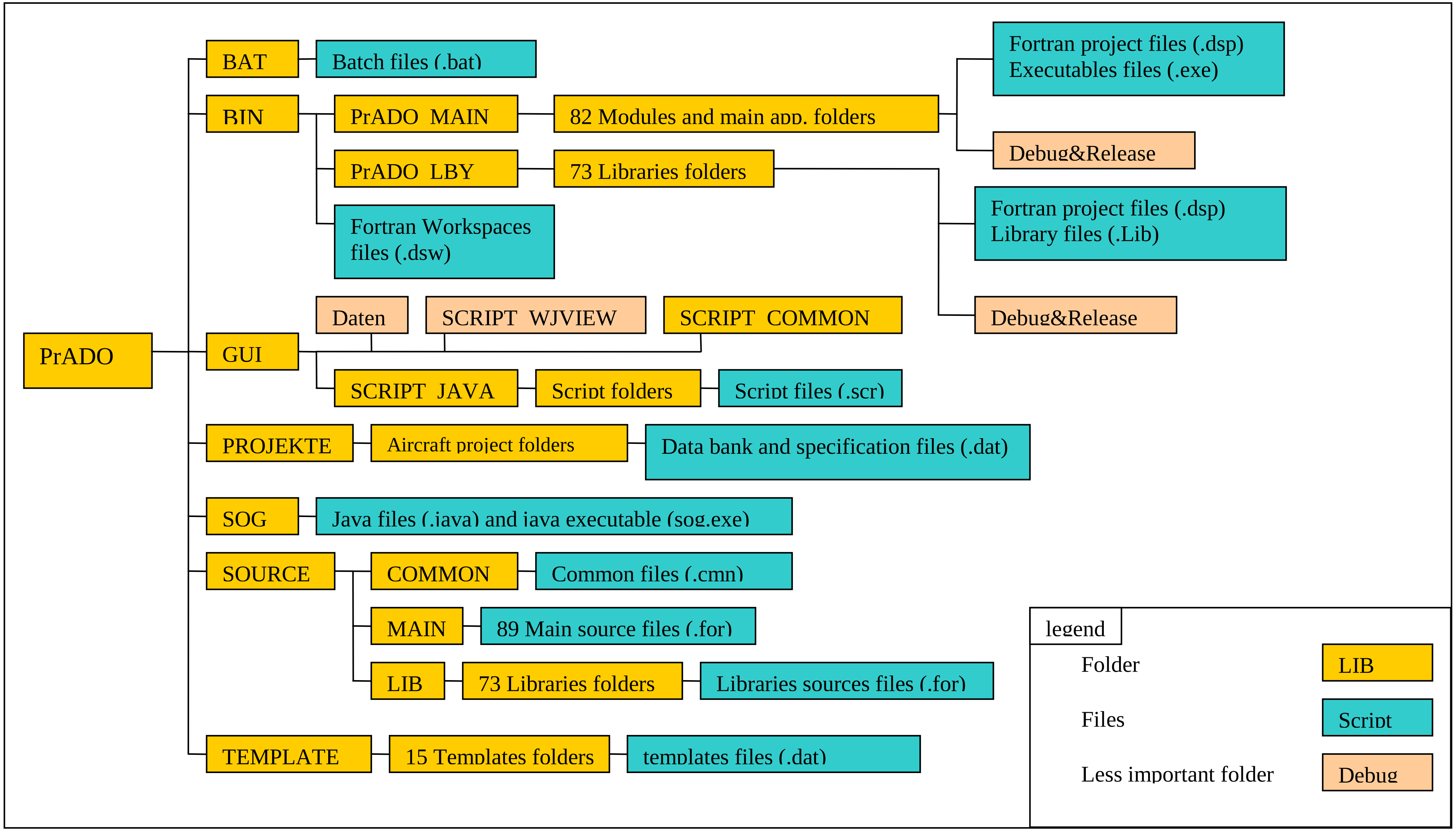

Figure 3.2 Tree of most important folders and files

The following schema explains the interactions and functions of the different folders

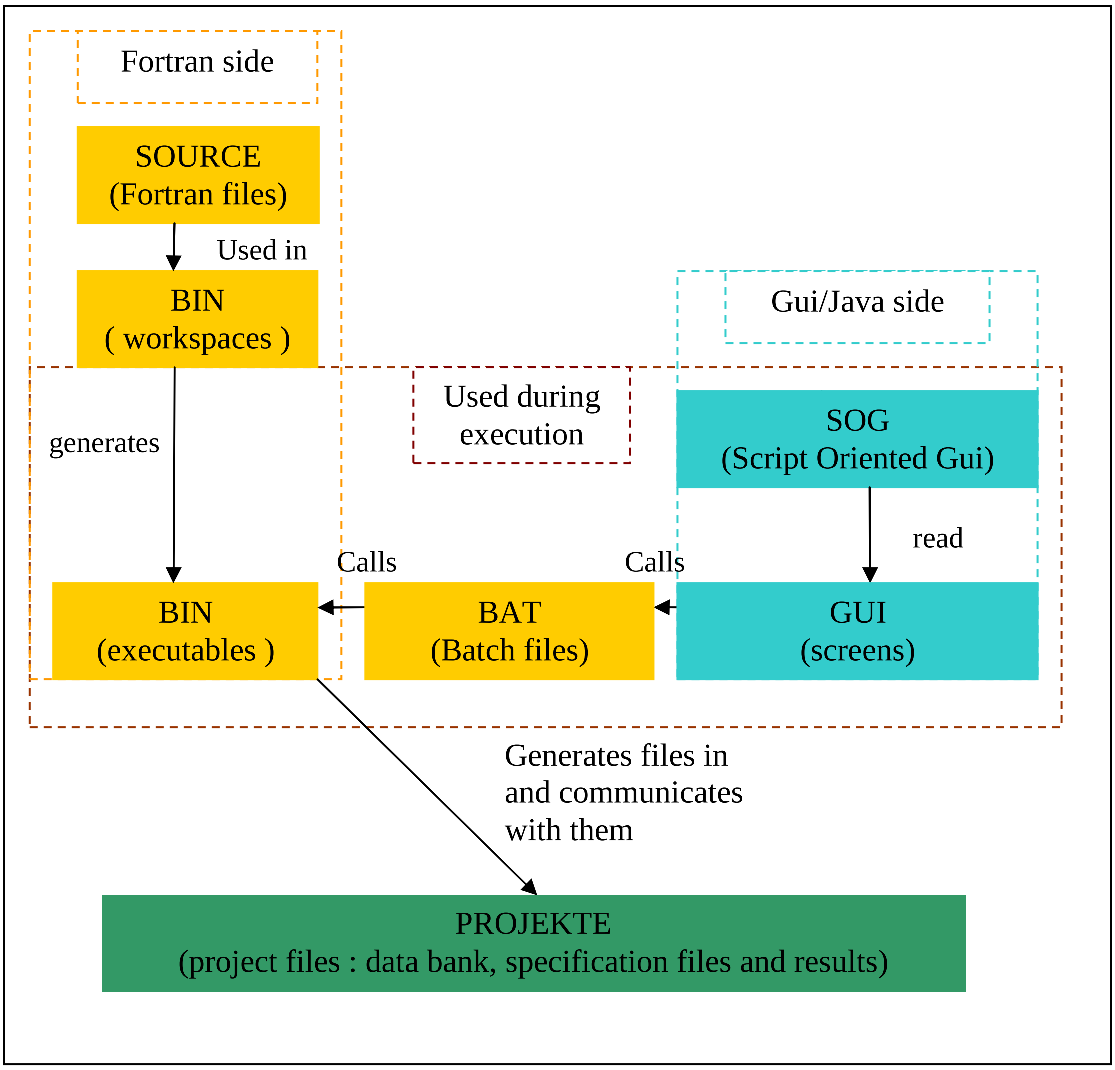

Figure 3.3 Schema of interaction between files by folders

A description of each folder can be given as follows:

In PRADO/BAT

It contains macro or batch files launching executable files compiled from the workspace files (.dsw and .dsp) from the PRADO/BIN folder. Those files can be executed by clicking on them in windows and are called by the GUI coded in JAVA. For example, clicking on the md1 button launches the md1.bat, which launches md1.exe, (in SOURCE/MAIN/md1/md1.exe).

There area total of 200 batch files 100 are for the Modules, and the others for graphical display or main options. Here is an example of what they look like. The only difference with other batch files is in the name of the executable launched.

```
@ECHO off
:: -----------------------------------------------------------
:: File: MD1-E.BAT
::
::      Batchdatei zum Ausfuehren des Programms MD1
::
:: -----------------------------------------------------------

:: Springen in das Arbeitsverezeichnis
cd %PrADOHOME%

:: Starten des Anwendungsprogramm
START %PrADOHOME%\BIN\PrADO_MAIN\MD1\md1.exe -PAUSE=JA –
SPRACHE=ENGLISCH

:: -----------------------------------------------------------
```

As one can see, the %PrADOHOME% directory is known: it is entered as variable at the installation of PrADO (see **KIESEL 2007**) and language options are specified here.
These batch files can be modified and read by right clicking on them in windows, and clicking editing. This will automatically launch the notepad and the text of the batch file within.
Calling batch files from the GUI to launch executable files is only a solution to the difficulties of launching directly executable files from the GUI.

In PRADO/BIN

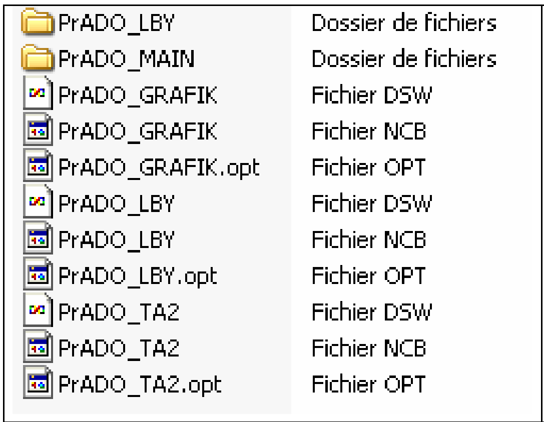

Figure 3.4 Sample of content of the bin folder

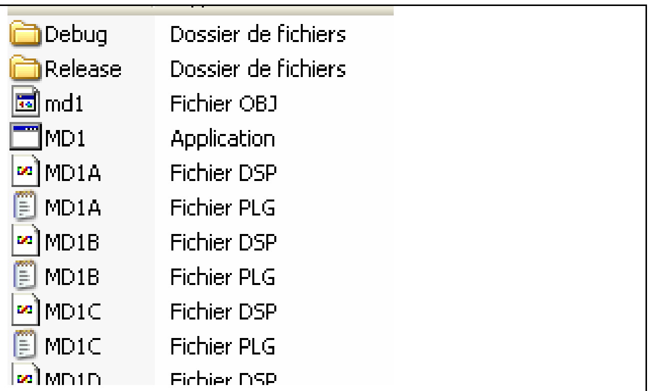

Figure 3.5 Sample of content of the Bin/PRADO_MAIN/MD1 folder

Here are ".dsp" and ".dsw" files which can be opened with the Fortran compiler (Microsoft Compaq Visual Fortran). They are Fortran workspaces and projects. This is a typical mechanism of Microsoft Visual Studio and compiling C++ project is very similar to this procedure. Workspaces are created; they include one ore more projects. Each project includes also one or more source file, external files and libraries. Source files, project files and workspaces don't have to be in the same folder as long as the compiler knows where the files are through the properties of each file in the workspace.

From a workspace, the possibility is given to compile one or all projects.

By opening a pre-existent workspace, one can change the Fortran files it involves. Those files are not located in the BIN folder but in the SOURCE folder as will be further explained later.

The sub-folder of the MAIN folder have executable files (.exe) generated by compilation of the Fortran file they contain.

On the other hand, in the sub-folder of the LIB folder, library files (.lib) generated by compilation of the Fortran file are to be found. Those files are used in other projects to dispose of all the functions defined in the libraries.

It is important to know that most of the codes doing the computation are not located in the project of the main folder but in the libraries.

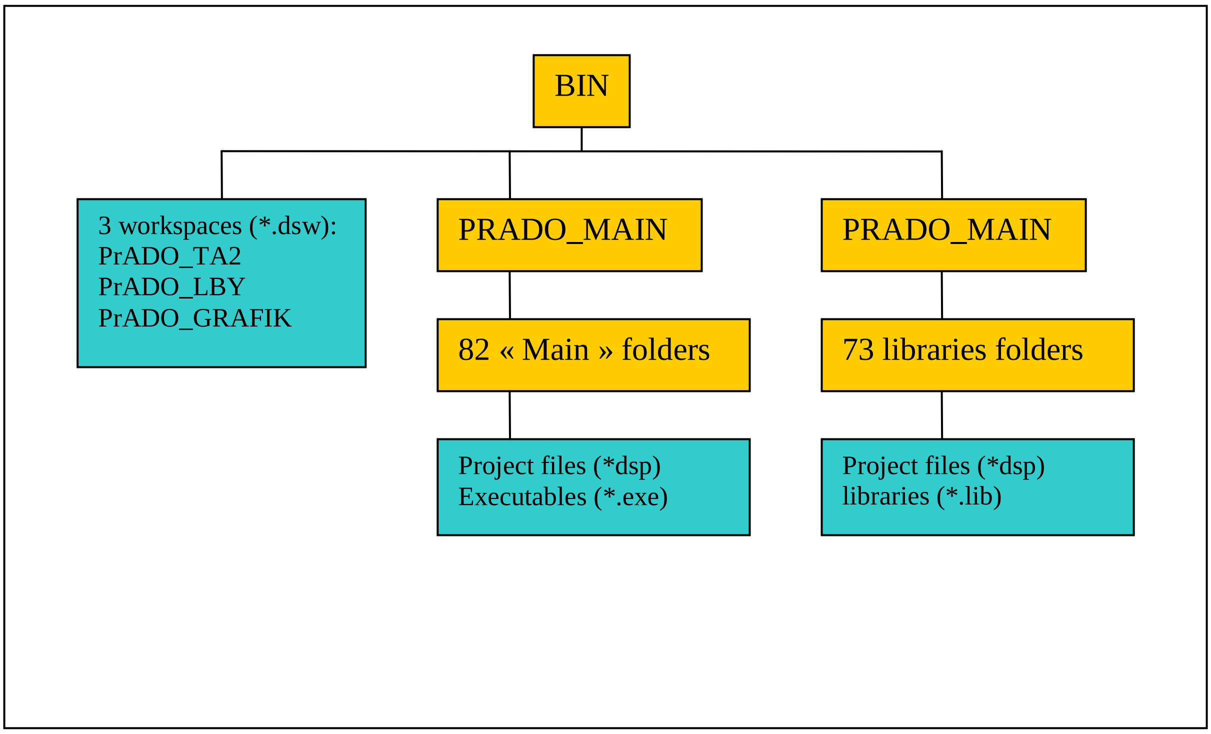

Figure 3.6 Structure of the PRADO/BIN folder

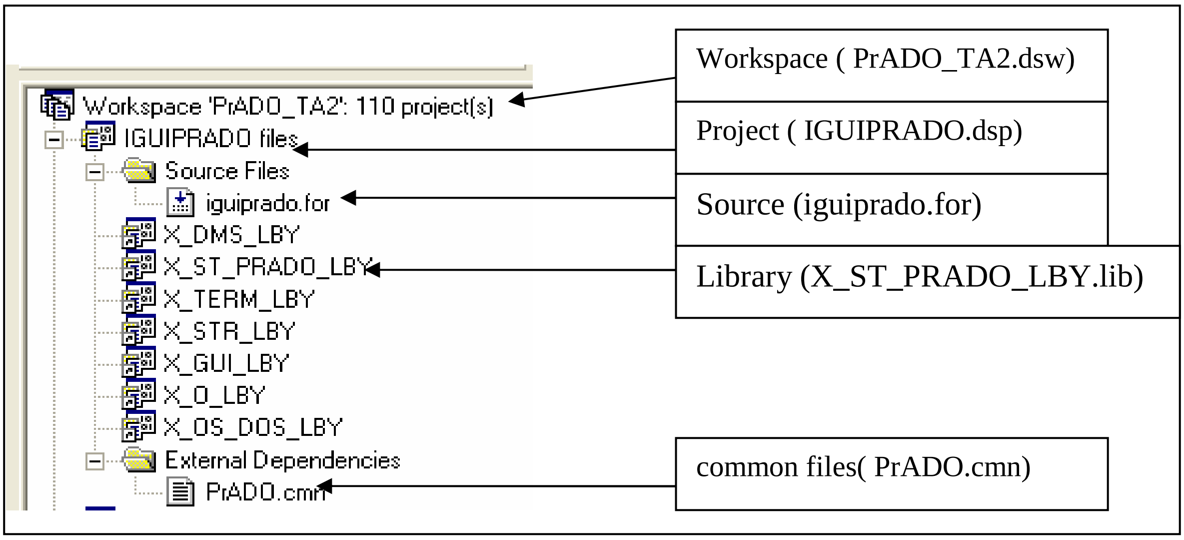

Figure 3.7 Files related to the Fortran compiler

In PRADO/PROJEKTE/PrADO-TA2/

Here are the project folders in which are the data banks generated by PrADO. The “Master input file» or specification file is also here with other folders belonging to each project.

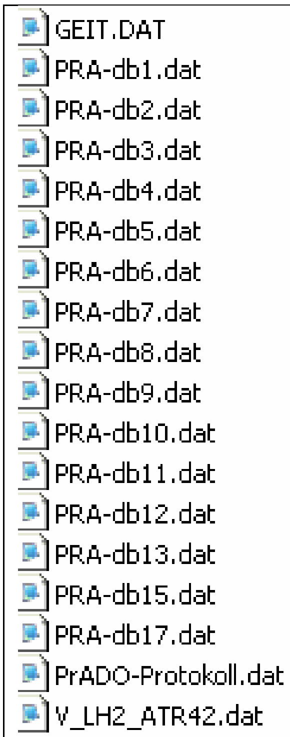

Figure 3.8 Content of a project folder

Data bank - Nr.	Field
DB 1	Software control
DB 2	Specification
DB 3	Wing
DB 4	Fuselage
DB 5	Horizontal tail
DB 6	Vertical tail
DB 7	Landing gear
DB 8	Propulsion
DB 9	Aerodynamic
DB 10	Masse/Gravity center
DB 11	DOC
DB 12	Flight mechanic
DB 13	LH2-System
DB 14	Fairing
DB 15	Aircraft Geometry
DB 16	Winglet
DB 18	External Tanks
DB 19	statistic specification

Table 3.1 List of the data bank

In PRADO/GUI:

There are 4 folders: SCRIPT_JAVA, SCRIPT_WJVIEW, SCRIPT_COMMON, and Daten. The SCRIPT_JAVA and SCRIPT_WJVIEW folders are quite similar. They contain scripts (.scr files) which describes the executable or the interface every button calls. In "SCRIPT_JAVA", they call batch files from the BAT folder and in the "SCRIPT_WJVIEW" folder they call link files (.lnk). The examination of the properties of the link files which launch the complete PrADO (PrADO TA2-A.lnk) shows that it executes the following line:

```
C:\WINDOWS\system32\javaw.exe -cp %PrADOHOME%\SOG sog
%PrADOHOME%\GUI\SCRIPT_JAVA\PrADO-TA2\PrADO-TA2-GUI-A.SCR
```

So from there, it can be observed that three different things are executed: the java machine (javaw.exe) but also the "sog.exe" executable and the script PrADO-TA2-GUI-A.SCR. From that and the explanations given by Mr. Heinze, it can be deduced that the script used for the version of PrADO which is described is in the"SCRIPT_JAVA" folder. The folder with the link files is useful in another version of PrADO.

The presence of the %PrADOHOME% variable, which is defined at the installation as said in **KIESEL 2007,** is to be outlined.

In PRADO /SOG:

SOG stands for "Script Oriented GUI".

Here are the java files used for the GUI (graphical user interface). Those files describe the basic shell of the interface but in fact they just call the real descriptions located in PrADO/GUI/…

In addition to the java files, there is also the executable (sog.exe) resulting from compilation of those files.

In PRADO/SOURCE:

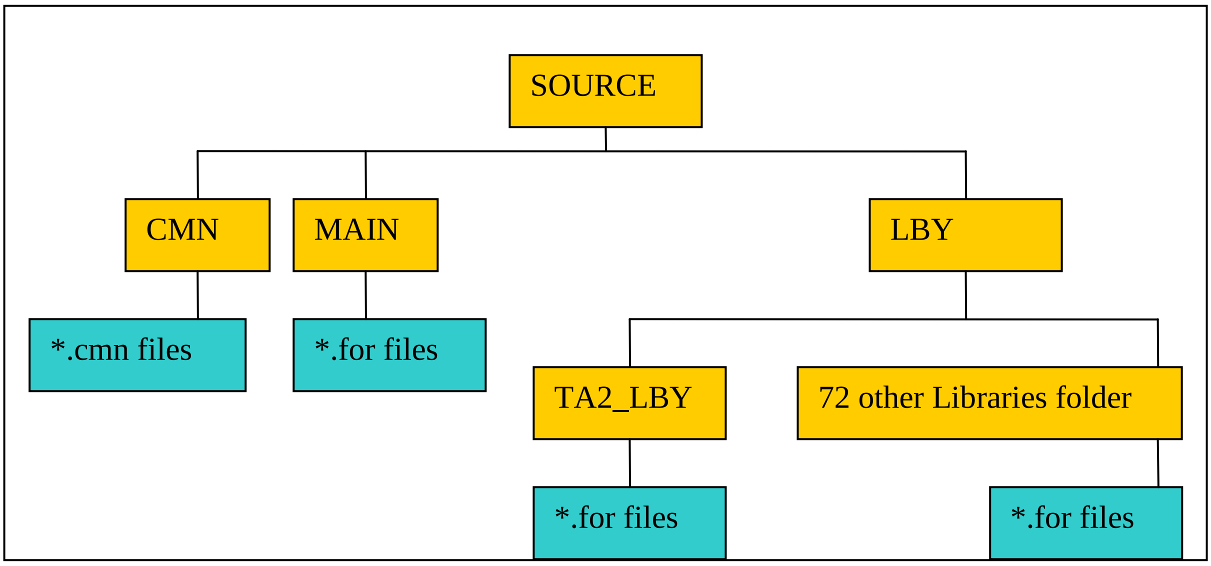

Figure 3.9 Structure of the PRADO/SOURCE folder

There are three folders in PRADO/SOURCE: MAIN, LIB and COMMON

- CMN is a set of “.cmn” files included by the Fortran subroutines. They contain variables used commonly by the Fortran programs, so that one line (include *.cmn) creates the variables contained in the .cmn file. They are equivalent to header files in C++. (CMN stand for common)

- MAIN contains the 89 Fortran files corresponding to the project of the Fortran executable projects of BIN/PrADO_MAIN.

- LBY contains 73 folders including the 1600 Fortran files related to the 73 Fortran libraries project located in BIN/PrADO_LBY/

The most important library is TA_LBY as explained later.

3.2. Relational diagrams

The following diagram shows the relations between files when PrADO is executed.

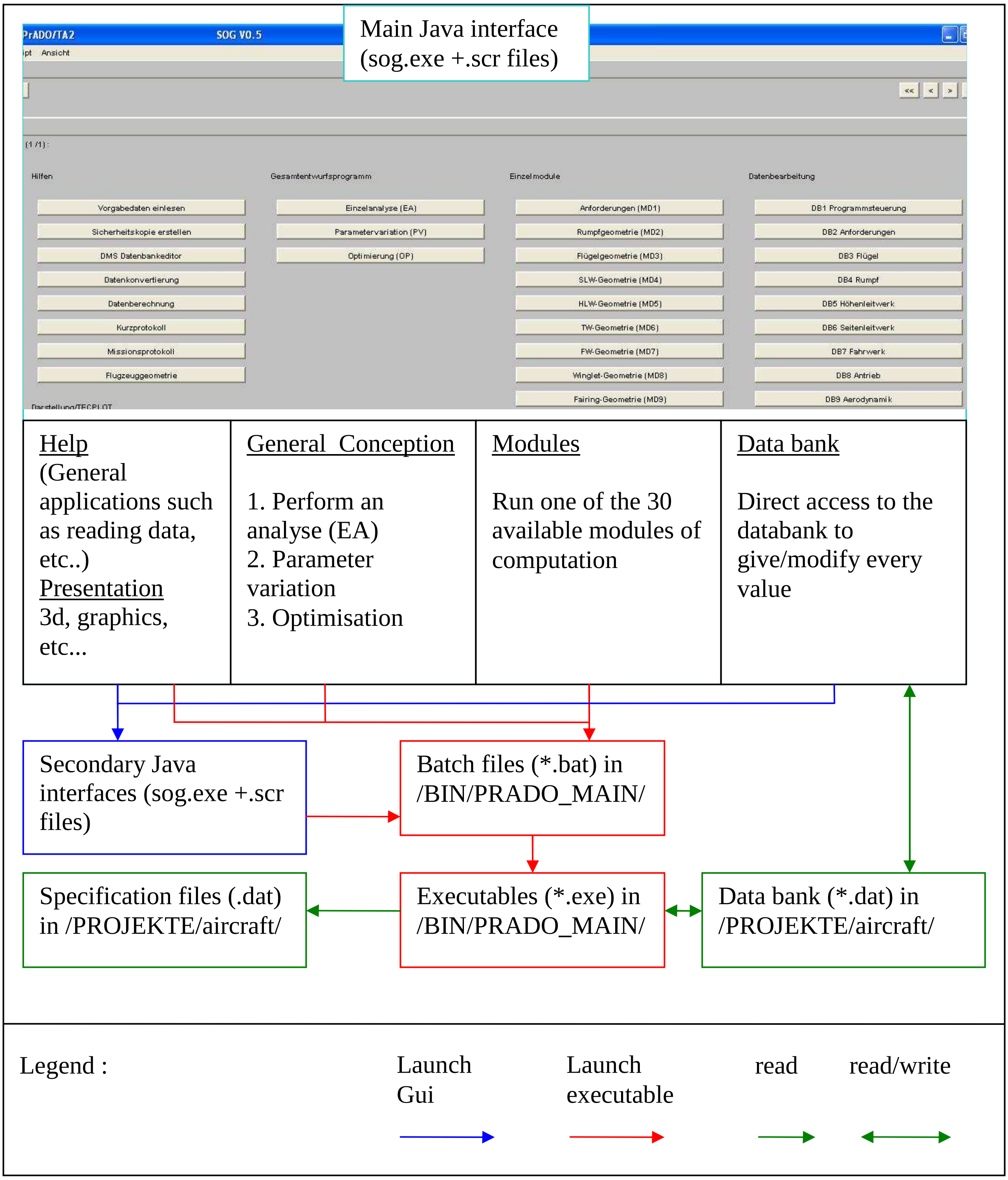

Figure 3.10 Relational diagram during execution of PrADO

The following table shows which source file generates which other file:

Source File	Generating software	Generated files
Java file (*.java) in /SOG/	Java compiler (refer to IFL for more information)	Executables (*.exe) in /SOG/
Source file (*.for) in SOURCE/MAIN/	Microsoft visual Fortran Compiler	Executables (*.exe) in /BIN/PRADO_MAIN/
Source file (*.for) in SOURCE/LBY/	Microsoft visual Fortran Compiler	Libraries (*.lib) located in /BIN/PRADO_LBY/ Which perform particular computation
Specification files (.dat)	PrADO	Data banks

Table 3.2 Generation of PrADO files

3.3. Modules

Modules are the basic computation unit. They will be launched one by one when a single analysis is performed, but can also be executed individually by clicking in the module section of the main interface.

The analysis of the code of those modules (in /SOURCE/MAIN/)shows that they all have the same structure.

A typical kind of file found in /Main is md6. It can be resumed as follow:

1. Declaration
2. Reading program options
3. Reading databank
4. Call the module/ CALL MD6
5. Set a flag stating that the module worked well or not
6. Generates a list of the in- and output values
7. Saving databanks
8. Pause

Complete code can be found in appendix MD6

It can be seen that MD6.exe runs a subroutine also called MD6. So it could be thought that the routine calls itself, but its is in fact calling another subroutine called MD6, which is located in TA2_LBY.

SUBROUTINE MD6

Algorithm:

```
-------------------------------------------------------------------
 Jet engines geometry (control program)
 -------------------------------------------------------------------
Description of input:
Description of output:
1. Declaration
2. Information on module
3. Data bank used
4. Reading necessary data from the data bank
5.  Installation of the propulsion system
        - Jet engine's conception-
      CALL MD6A - (thermodynamic Data, Dimensions, weight and gravity centre)
      CALL MD6B - Geometry, Arrangement, Weight and gravity centre of the jet engine
      pylon-
DO two times
      CALL MD6C Geometry, Arrangement, weight and gravity centre of the jet engine
      nacelle-
      CALL MD6D Arrangement, weight and gravity centre of the jet engine-
6. Automatic generation of a 3D-propulsion-system
```

The complete code can be found in **appendix B**

So it can be seen that it launches four subroutines beginning by md6 : MD6A, MD6B, MD6C, und MD6D. Each of those files performs a specific computation about jet engines. Those subroutines are also located in TA2_LBY. For more information about this module, see the chapter about jet engine conception.

So this is how PrADO modules work: a generic code calls another main subroutine which calls more specific subroutines. A look at the description and structure of those modules will help a lot for the understanding of PrADO:

MD1: Specification

MD1 checks and completes the specifications as well as the statistics parameter for initial values of the conception parameter for the most important design computation, which will be exactly determined through the iterative process. (Control program)

Algorithm:

```
    SUBROUTINE MD1
1.  Declarations
2. Information on module
3. Data bank used:
4. Read necessary data from the data bank
  - Read specification of aircraft configuration-
5. Choice of the computation model
  - If the aircraft is a conventional tail aft aircraft -
        Then CALL MD1A1
             CALL MD1A2
  - If the aircraft is a BWB or flying wing -
        Then CALL MD1A1
        CALL MD1B2
  - If the aircraft is a three-surface-aircraft -
        Then CALL MD1A1
        CALL MD1A2
  - If the aircraft is a Canard type -
        Then CALL MD1A1
        CALL MD1A2
  -If the aircraft is an unmanned tail aft aircraft -
        Then CALL MD1C1
        CALL MD1C2
  -If computation model is unknown -
    ELSE CALL TA2ERR1 (…)
```

MD2: Fuselage geometry

MD2 is the control program for the fuselage geometry: the specified fuselage type is defined here to the program and subroutines called here perform the geometry computation.

Algorithm:

```
1.  Declarations
2. Information on module
3. Data bank used:
4. Einlesen der erforderlichen Daten aus den Datenbanken
5. Choice of the computational model
  - If the aircraft is a conventional tail aft aircraft, a three surface aircraft, a canard aircraft -
        Then CALL MD2A1
         Else CALL MD2A2
  - If the aircraft is a BWB or a flying wing-
```

CALL MD2A1
CALL MD2B
ELSE CALL MD2A2
- If the aircraft is an conventional unmanned tail aft aircraft-
CALL MD2A2

MD3: Wing geometry

Algorithm:

1. Declarations
2. Information on module
3. Data bank used:
4. Einlesen der erforderlichen Daten aus den Datenbanken
5. Choice of the computation model

- If the aircraft is a conventional tail aft aircraft, a three surface aircraft or a canard aircraft-
Then CALL MD3A

- If the aircraft is a BWB or a flying wing-
Then CALL MD3A
CALL MD3B

MD4: vertical tail, rudder geometry

Algorithm:

1. Declaration
2. Information on the module
3. Check if a data protocol is defined
4. Constant, no-go criteria
5. Databank used
6. Description of the data bank configuration
7. Reading the data needed from the data bank

- Program control–
- Wing geometry –
- Fuselage geometry –
- Horizontal tail plane - Geometry –

8. Display the inputs
9. Computation model
    9.1 Build the profile data bank and save it
    9.2 Read the vector with the position of the vertical tail plane
    9.3 Compute the plan geometry
        9.3.1 Build the allocation vector
        9.3.2 Span width
        9.3.3 Estimation for the roots depth and save the reference taper ratio

9.3.4 Data record for the vertical tail plane geometry and save in the data bank
9.3.5 Root depth
9.3.6 External depth
9.3.7 Taper ratio (TAS/TIS)
9.3.8 Geometrical Reference depth
9.3.9 Position of the gravity centre of the horizontal tail plane, of the geometrical angle of attack, and position of the rotation point
9.3.10 ETA coordinates of the tail plane attachment
9.4 Position of the coordinate reference system
9.5 ETA-node (Vector)
9.6 Position of the geometric neutral point
9.7 Wetted area
9.8 Middle maximal relative thickness
9.9 Middle sweep of the leading edge
9.10 Middle sweep of the 25%-Line
9.11 Middle sweep of the 50%-Line
9.12 Middle sweep of the trailing edge
9.13 Middle dihedral configuration of the 25%-Line
9.14 Data record for the vertical tail plane construction and save of the data bank

9.15 Maximum available tank volume, volume and position of the gravity centre of the individual tanks
9.16 Data record for the fixed leading edge elements develop and in the data base store
9.17 Data record for the trailing edge elements develop and in the database
9.18 Data set for the form's construction and save in the data bank
9.1000 Vertical tailplane geometry is processed
10. Display results
11. Automatic generation of a 3D vertical tail plane

***MD5**: Horizontal tail plane geometry*

Algorithm:

1. Declaration
2. Information on modle
3. Check if a data protocol is defined
4. Constant, no-go criteria
5. Data bank used
6. Description of the data bank configuration
7. Read necessary data from the data bank
8. Display results
9. Computation part

MD7: Landing gear geometry

Algorithm:

1. Declaration
2. Information for program module
3. Check if a data protocol is defined
4. Constant, no-go criteria
5. Data bank used
6. Description of the data bank configuration
7. Read necessary data from the data bank
8. Display results
9. Computation part
    - 9.1 Initial values specify
    - 9.2 Montage on fuselage
    - 9.3 Montage on the wing
    - 9.100 Save results in Data bank
    - 9.101 Save results in Data bank
    - 9.102 Global Geometry data of the landing gear
10. Results
11. Automatic generation of a 3D-landing gear - Presentation

MD8: Winglet geometry

This module is having the same algorithm than the horizontal and vertical tail plane module.

MD9: Fairing geometry

Algorithm:

1. Declaration
2. Information on module
3. Data bank used
4. Read data needed from the data bank
5. Delete data, which this subroutine will recalculate
    - -fuselage geometry
    - -masse and gravity centre of the fairing
    - -Design of the fairing
    - CALL MD 9A

MD10: Geometry of the LH2 systems

Algorithm:

1. Declaration
2. Information on module
3. Data bank used

4. Reading the data needed from the data bank
5. Design of the LH2 systems: -Pylon: CALL MD10A -Tank with isolation: CALL MD10B 6. Automatic generation of a 3D model of a 3D tank with its pylon

MD11: Geometry of the aircraft

Algorithm:

1. Declaration
2. Information on module
3. Data Bank used
4. Reading the data needed from the data bank
5. Computation model:

   - If the aircraft is a conventional aircraft, a three surface aircraft a multiple fuselage aircraft or a canard aircraft then CALL MD11A
   - If the aircraft is a BWB or a flying wing then CALL MD11B
   - If the aircraft is an unmanned conventional aircraft then CALL MD11C:

MD12: Aerodynamic (control program)

Algorithm:

1. Declaration
2. Information on module
3. Data bank used
4. Reading necessary data from the data bank
5. choice of the computation model : CALL MD12A

MD13: Installation of the aircraft system

Algorithm:

1. Declaration
2. Information on module
3. Check if a data protocol is defined
4. Data dank used
5. Read data
6. Results for control
7. Leading the computation model

7.1. Results for control

7.2. Installation of the air conditioning:

CALL SYS_ECS_GEN

7.3. Electrical power supply

CALL SYS_ELE_GEN

7.4. Flight control

CALL SYS_FCS_GEN
7.5. Hydraulic power supply
CALL SYS_HYD_GEN
7.6. Landing gear
CALL SYS_LDG_GEN
7.7. LH2-System
CALL SYS_LH2_GEN
7.8. LCHx-System
CALL SYS_LCHX_GEN
7.9. Network installation
7.9.1. Computation of the requirements
CALL SYS_NET_GEN
7.9.2. Computation of the concentrated elements
CALL SYS_GEN_GEN
7.9.3. Computation of the properties
CALL SYS_NET_CAL
7.10. Power withdraw of the jet engine
CALL SYS_GEN_PWR
8. Monitoring the necessary data from the data bank
9. Display the results

MD14: Installation of the propulsion system

Algorithm:

1. Declaration
2. Information on module
3. Check if a data protocol is defined
4. Constant
5. Data bank used
6. Description of the of the calculated data
7. Reading necessary data from the data bank
8. Control of value
9. Computation part
   9.0. Fixing the first value and general data
   9.1. Estimation of the critical jet engine in case of jet engine failure
   9.2. Estimation of the thrust need for different flight cases
   - Take-Off with One Engine Out
   - Beginning of Cruise (Mission with Maximum Payload)
   - End of Cruise (Mission with Maximum Payload)
   - End of Cruise (Mission with Maximum Fuel)
   - Landing Approach with One Engine Out
   -Touch-and-Go during Landing

- Beginning of cruise with one Engine out (Mission with Maximum Payload)
- Beginning of Cruise with One Engine out (Mission with Maximum Fuel)

9.3. Estimation of the thrust need according to take off length

9.100. Estimation of the thrust of all jet engines

9.101 Specific fuel combustion in different flight phase (with all jet engines similar)

9.102 Specific fuel combustion in different flight phase (with every jet engine considered particular)

9.103 New value for the parameter IBATW1

10. Display the results

The chapter about jet engines contains more details upon this subject.

3.4. Global Analysis of the Libraries

73 libraries are located in PrADO/SOURCE/LIB/. Each library has a particular purpose: Definition of aircraft part geometry, communication with Tecplot or data bank, computation of performance. Here comes a list sorted by the sections they belong to:

Modules libraries (3^{rd} level)	various libraries	Aerodynamic libraries (4^{th} Level)	Aircraft Conception libraries (4^{th} Level°
TA1_LBY	ALLG_LBY	AERO1_LBY	FST_SYS_LBY
TA2_LBY	GUI_LBY	AERO2_LBY	DOC_LBY
	IF_MDi_LBY	AERO3_LBY	FAR_LBY
	LAST_LBY	AERO4_LBY	FW_LBY
	O_LBY	AEROH_LBY	IFL_SYS_LBY
	OS_DOS_LBY		JAR_LBY
	REIN1_LBY_		SPBE_LBY
	SENS_LBY		FM_LBY
	STR_LBY		FLIGHT_LBY
	TEC_LBY		
	TERM_LBY		
	KONV_LBY		

Jet engine libraries (4^{th} level)	Weight libraries (4^{th} level)	Mathematical libraries (2^{nd} level)	Data base libraries (1^{st} level)
TW1_LBY	WEIGHT1_LBY	EX_LBY	DMS_V1_LBY
TW2_LBY	WEIGHT2_LBY	MATHE_LBY	DMS_V2_LBY
TW3_LBY	WEIGHT3_LBY	ITPROG_LBY	
TW4_LBY	WEIGHT4_LBY		
TW4_LBY			
TW5_LBY			
TW6_LBY			
GEOTW1_LBY			

Geometrical libraries (1^{st} level)		
2DKURVE_LBY	GEOFL1_LBY	GEORF1_LBY
2DLFZ_LBY	GEOFW1_LBY	GEOSL1_LBY
3DAIRPORT_LBY	GOBJ1_LBY	GEOTW1_LBY
3DBFZ_LBY	GEOHL1_LBY	GEOWL1_LBY
3DMENSCH_LBY	GEOLHTK1_LBY	IOLIFTINGLINE
ADS_LBY	GEOLHTK2_LBY	GPY1_LBY
GEOEXTK1_LBY	GEOQ1_LBY	GEOH_LBY
GEOFAIR1_LBY		

Table 3.3 Tables of libraries by section of concern

A common way to describe a program is the diagram showing its source code divided into the parts which deal with the graphical part, the parts which manage the whole system and the data. In most programs, all the different part would be :

Graphic		**Manager**	**Data**
Output :	**Interface :**	**Fortran Libraries**	**Specification file** **Data bank files**
Tecplot / Fortran Monitor	**Java**		

Table 3.4 Table of program by function

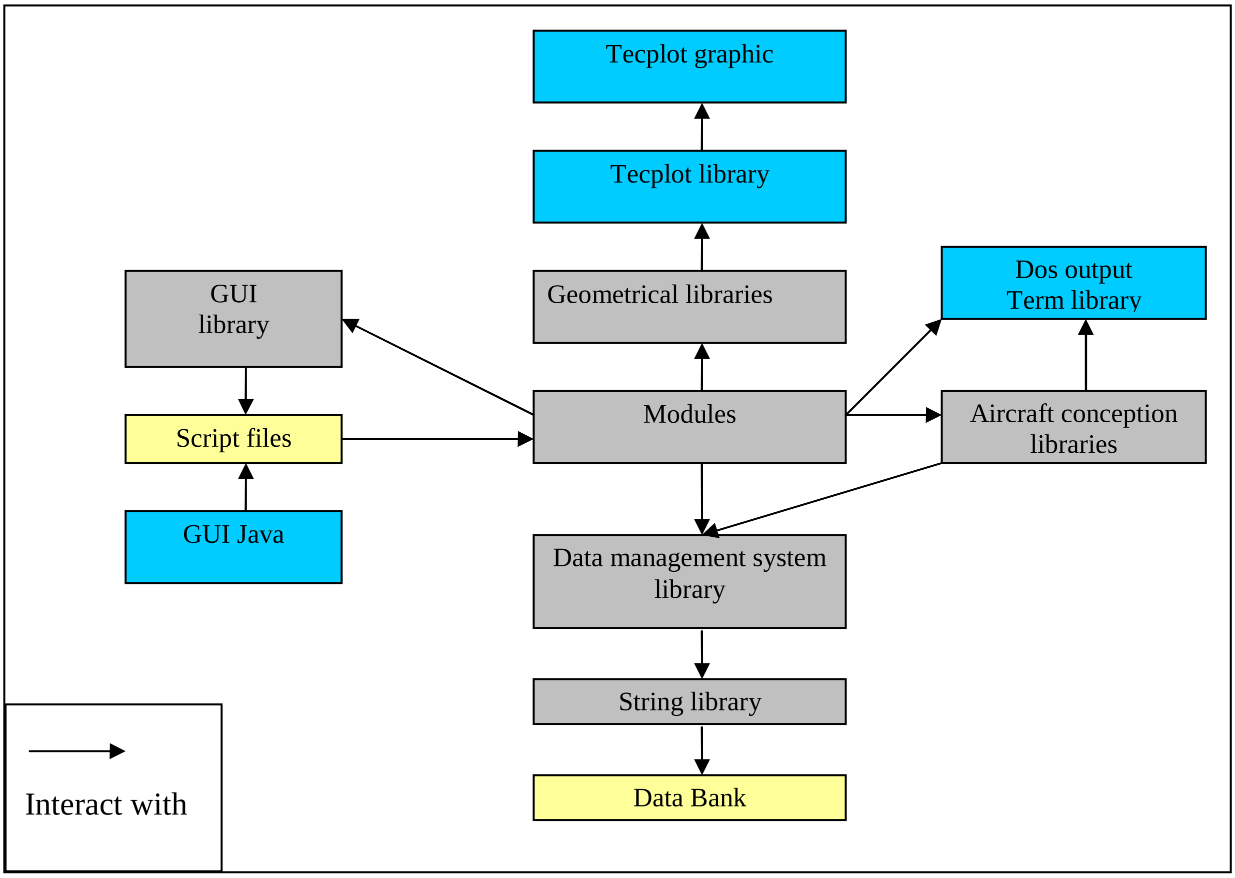

Figure 3.11 Interactions between libraries

3.5. Analyse of specific Libraries

Here comes a small description of the most important library found:

ALLG_LBY: ALLG stands for "Allgemein" meaning General library: it contains 7 commonly used subroutines such as **ATMOS, which** provide air properties at a given altitude, or **HOEHE,** which gives the altitude according to the air density.
FST_SYS_LBY: The aircraft system library provides the subroutines to compute the performance and characteristic of the aircraft systems, meaning the electrical system, the hydraulic system, the flight control and the net related to those systems.
IFL_SYS_LBY computes values for more aircraft systems such as the oxygen system, the APU, the anti-ice system.
DOC_LBY : Direct operational cost are computed by the subroutines of this library
FAR_LBY is the Flight aviation regulation library. It computes some values defined by the FAR25.
JAR_LBY is the Joint Aviation regulation Library. It computes values defined by the JAR22. Most of them concern gliders.
FW_LBY contains most of the landing gear subroutines, which will be called in the landing gear modules (MD7 and MD15)
SPBE_LBY provides programs which permit to compute the gravity centre for an aircraft configuration according to a given payload case.
FM_LBY is the flight mechanic library .Its subroutines computes value such as the range, the cruise speed or maneuverability of the aircraft.
FLIGHT_LBY is the flight simulation library. Its subroutines simulate all kind of flight missions.
GUI_LBY is the Graphical User Interface library. As already seen, the interface part of the program is taken in charge by a java program, but this library permits the transfer of data from the program to the data banks.
KONV is the converting library. The subroutines in it permit the translation of files from one format to another one. This is how the interface to other program such as Tecplot or Nastran is made.
O_LBY is an Output library. It presents results from particular requests (Optimization, 3d or 2d graphical generation, List of variables implied, etc...)
OS_DOS_LBY is only one fortran file which contains 17 subroutines allowing to perfom DOS operations such as opening a file, reading it, getting its name... It also has 10 useful functions which give the time, the date , etc...
REIN1_LBY is a pack of subroutines helping to model an aircraft cabin
SENS_LBY is the sensitivity library.

STR_LBY is the string management library. The subroutines of the data management system use them a lot.
TEC_LBY is the Tecplot interface library. Some of them generate files which can be interpreted by Tecplot as graphic, 2d or 3d Model, e.g. some of them just give titles to the Tecplot graphics.
TERM_LBY is the monitoring library. It permits to output lines on the output windows. It contains the subroutine equivalent to the "cout" in C++ or the "System.out.println()" in Java

Table 3.5 Descriptions of specifics libraries

4. Description of the Data Management System Library

A particularity of PrADO is its data bank system. For each project of aircraft design, 19 data bank files are created, which contain all variables related to the aircraft and its design. Here again, more information can be found in **Kiesel 2007,** which examine in details the database.

The connection between the Fortran subroutines and those data bank files are made by a specific library of subroutines located in source/lib/dms_lby and in the latest release, a second library is added: source/lib/DMS_V2_LBY. No relevant difference has yet been found between the first and second library, except that in the latest release the folder DMS_V1_LBY doesn't have the subroutine db_h2 even if the first release had it and the DMS_V2_LBY too.

The report will only describe DMS_V2_LBY, as it is the last release of the library. It contains 35 Fortran files. They all begin with "db_...", and all include the "dms.cmn" file. They realize different function such as: saving a specific type of variables, reading another, checking all variable of a databank, editing one other.

.

Here follows a list of every subroutine of the DMS_V2 library and their descriptions:

db_a6
This subroutine sets all control parameters, which define a variable as optimization size, back to the value 0.
db_cmd
This subroutine examines the program call after relevant options and converts these in control parameters and input data for the DMS.
db_del
This subroutine deletes completely the indicated variable from the database.
db_err1
This subroutine sends warnings and error messages

db_h1
This subroutine calls a variable from the data bank. The variable can be a character, an integer or a real. If the variable does not exist in the data bank, then a return happens without abnormal termination.
db_h2
This program gives a list with the variable, which will be used in a module. Those variables are divided into inputs and outputs variables.
db_r0
Calls of the maximum number of the specified data bases.
db_r1
This program checks how many variables are saved in the NDB data bank.
db_r2
This subroutine checks if the variable already exists in the data bank.
db_r3
This subroutine gives the description of the variable.
db_r4
This subroutine gives the unit of the variable
db_r5
This subroutine determines the field sizes of the variable. If this variable is missing in the database, the program stops.
db_r6
This subroutines tells if the variable is an optimisation variable or not.
db_r7
This subroutine gives the type of number of the variable.
db_r8
This subroutine checks, which variable name has the variable specified by the IVAR number. This variable will be researched in the data bank NDB.
db_r9
This subroutine determines the field sizes of the variable
db_rc8
This subroutine calls an Integer variable from the data. If this variable is missing in the database, the program stops.
db_rc9
This subroutine calls an Character variable from the data , if this variable is missing in the data base, the program send an error code (0= available, 1= missing)
db_ri8
This subroutine calls an Integer variable from the data, if this variable is missing in the database, the program stops.
db_ri9

This subroutine calls an Integer variable from the data , if this variable is missing in the data base, the program send an error code (0= available, 1= missing)
db_rr8
This subroutine calls a REAL*8 variable from the data bank. If this is missing from the data bank, the program stops
db_rr9
This subroutine calls a REAL*8 variable from the databank. If this variable is missing in the data base, the program sends an error code (0= available, 1= missing)
db_s6
This subroutine saves a control parameter IOP stating if the variable is an optimisation variable in the data bank. If the variable does not exist in the data bank the program stops.
db_sc1
This subroutine saves a complete data set of type Character in the database.
db_sc8
This subroutine saves only the data value of an integer value in the database. If the variable is missing in the database, the program breaks off.
db_si1
This subroutine saves a complete data set of type integer in the database.
db_si8
This subroutine saves only the data value of an integer value in the database. If the variable is missing in the database, the program breaks off.
db_sr1
This subroutine saves a complete data set of type real *8 in the data bank.
db_sr8
This subroutine saves a complete data set of type real *8 in the data bank. If the variable is not in the data bank, the program stops.
db_sav
This subroutine creates a back up of the database or reads one.
dmscl2
This subroutine closes the data base and creates a file for every data base
dmscl3
This subroutine closes the data base and creates a general file for all the data base
Dmsed
This subroutine calls a data editor.
Dmsedd
This subroutine is a data editor, with which the data in the databases can be observed and changed. The communication will use the standard input and – output canal.
Dmsede

This subroutine is a data editor, with which data can be observed and changed in the databases. For the dialogue the standard input and - output canal will be used

Table 4.1 List of the subroutines of the DMS_V2 library and their descriptions

The subroutines mostly used in this library are the ones saving and calling variables.
Two good examples are for instance db_ri8 and db_si8, which have the following structure:

db_ri8 is an example of a subroutine reading a variable from the data bank

Declaration Check if the variable is already saved in the data bank Variable does not exist in the data bank (Program stops) Variable exists - Variable information - checks if the number type is correct - checks if the number dimension is correct Display results

db_si8 is an example of a subroutine saving a variable in a data bank

Declaration Check if the variable is already saved in the data bank Variable does not exists in the data bank (Program stops) Variable exists - Old variable is overwritten

Those two examples use two subroutines: DB_ERR1 in case of program stop and STR41 if it has to communicate with the data bank. STR 41 comes from the String library STR_LBY and it might be useful to look at its description: this program determines the number of the first sign and of the last sign, which is not a blank for a given variable

Example:

VARIABLE = ' TEST ' (8 signs)

IA = 3 (number of the first sign which is not a blank)

IE = 6 (number of the last sign which is not a blank))

It gives the opportunity to locate the address of the variable in the data bank files and to make a copy of it character by character into the program variable.

Many files are interesting to quote here:
dms.cmn in /SOURCE/CMN/ gives a description of variables commonly used in all subroutines of the DMS library. Here comes a translated version of this description:

Variable	Description
NDBMAX	maximum number of manageable data bases
VNAME (IP1)	vector with the variable names
VNAME (I):	Name of the variable n°I
VEIN(IP1)	the vector with the units related to the variables
VEIN(I):	Unit of variable n°I
VBES(IP1)	the vector with the descriptions to the variables
VBES(I)	Description of the variable n° I
VINFO(IP1, ip2)	memory array with information on the properties of the variables for the variable n°I applies
VINFO(I, 1)	Marking whether the variable is an optimization variable
VINFO(I, 2)	- Marking on the type of number = 1 CHARACTER type = 2 INTEGER type = 3 REAL type
VINFO(I, 3)	Marking of the type of variable = 1 scalar, vector = 2 matrix
VINFO(I, 4)	1.Dimension the variable
VINFO(I, 5)	2.Dimension the variable
VINFO(I, 6)	Number of the place in the hypervector, where stands the first element of the variable. The type of number determines the hypervector.
VINFO(I, 7)	Number of the data base, where the variable is put down
VINFO(1,8)	Counter, how often the variable was called up
VINFO(1,9):	Counter, how often the variable was stored
CHY(IP3)	hypervector with stored CHARACTER variables
IHY(IP4)	hypervector with stored INTEGER variables
RHY(IP5)	hypervector with stored material variables
VNAME1(IP1)	backup copy of the vector VNAME
VEIN1(IP1)	backup copy of the vector VEIN
VBES1(IP1)	backup copy of the vector VBES
VINFO1(IP1, ip2)	backup copy of the matrix VINFO
CHY1(IP3)	backup copy of the vector CHY
IHY1(IP4)	backup copy of the vector IHY
RHY1(IP5)	backup copy of the vector RHY
ICDMS1	channel number for input over keyboard
OCDMS0	channel number for display output
IDMSP1	control parameter/language with input and output 1: German 2: English
IDMSP2	control parameter/error handling 0: Error message is sent, without the program stops 1: Error message is sent and the program stops

Table 4.2 List of the variables of the dms.cmn file with their descriptions

5. The Jet Engines in PrADO

5.1. Jet Engines Modules

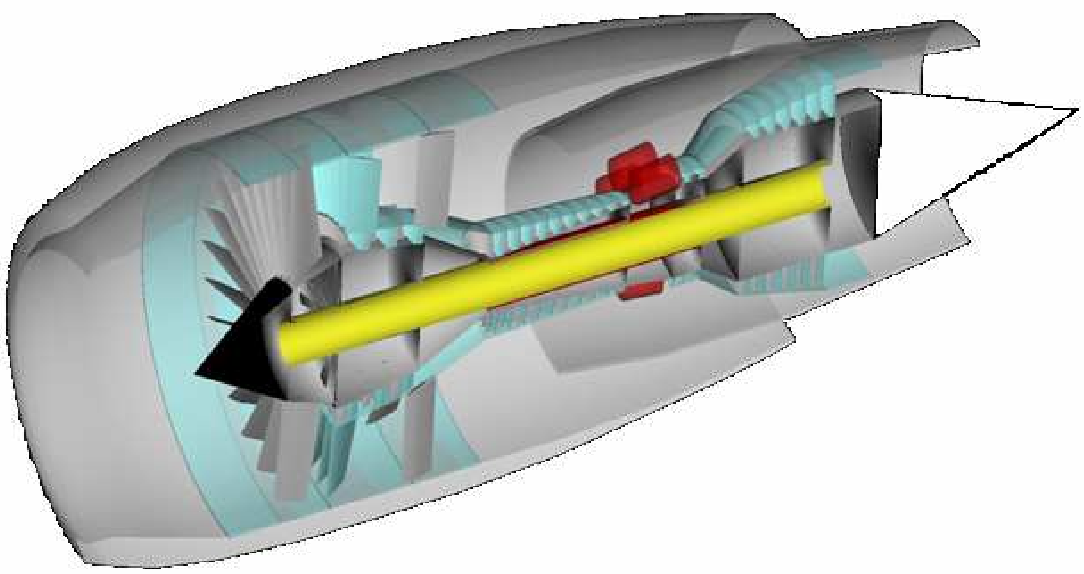

Figure 5.1 3D model of a jet engine designed with PrADO (from **IFL**)

Three modules deals with jet engine in PrADO: MD6, MD14, and MD33

MD6 In the latest version, MD6 call MD6A, B, C or D as already shown in chapter 3.2.

-MD6A computes thermodynamic data, jet engine dimensions, jet engine masses and centre of gravity. This algorithm is structured as follows:

```
------------------------------------------------------------------
 Jet engine conception
Calculation of:  thermodynamic Data
              o Jet engine data
              o Jet engine masse und gravity centre (LKS)
 ------------------------------------------------------------------
1. Declaration
2. Information on module
3. Check if a data protocol is defined
4. Constant
5. Data bank used
6. Reading necessary data from the data base
7. Monitoring the inputs
```

8. Computation part
    8.1 Fixing control parameters and initial values
    8.2 Computation model *** Jet engine is given***
    8.3 Computation model *** Model of HEINZE/DECHOW (1991) ***
    CALL TWHP1 (3, …) +CALL VTW1 (1,…)
    8.4 Computation model *** Model of HEINZE/DECHOW (1991) ***
    CALL TWHP1 (3,…) +CALL ZTL2G (…)
    8.5 Computation model *** Model of HEINZE/LEITNER (1999) ***
    CALL TWHP1 (3,…) +CALL ZTL10 (…)
    8.6 Computation model *** Model of MATTINGLY/HEINZE (FAN-LPC-HPC-HPT-LPT/2006) ***
    CALL TWHP1 (3,…) + CALL TWHP2 (1,…) +CALL ZTL11 (…)
    8.7 Computation model *** Model of MATTINGLY/HEINZE (FAN-HPC-HPT-LPT/2006) ***
    CALL TWHP1 (3,…) + CALL TWHP2 (1,…) +CALL ZTL12 (…)
    8.900 Computation model is unknown
    9. Display the results

MD6B computes the geometry of the jet engine pylon, its position on the aircraft, the masse of the pylon and its centre of gravity.

Geometry of the jet engine pylon, position on the aircraft, masse of a single pylon and gravity centre
----
1. Declarations
2. Information on module
3. Check if a data protocol is defined
4. Constant
5. Data bank used:
6. Description of the data set
7. Reading necessary data from the data bank
8. Monitoring the inputs
9. Computation part
    9.1 position of the pylon on the aircraft
    9.2 building the profile data bank save it in the data bank
    9.3 Geometry data of the pylon
    9.4 Mass and gravity centre of a single pylon
10. Monitoring the outputs

-MD6C computes the geometry of the jet engine nacelle, its position on the aircraft, the masse of the nacelle and also its centre of gravity.

Geometry of the nacelle, position on the aircraft, masse of a single nacelle and gravity centre

----

1. Declarations
2. Information on module
3. Check if a data protocol is defined
4. Constant
5. Data bank used:
6. Description of the data set
7. Reading the data from the data bank
8. Monitoring the inputs
9. Computation part
    - 9.1 Geometry data of the nacelle
    - 9.2 Position of the nacelle on the aircraft
    - 9.3 Geometrical data of the nacelle for aerodynamic computation
    - 9.4 Masse and gravity centre of the nacelle
10. Monitoring the outputs

-MD6D deals with the position of the jet engine on the aircraft, the masse of every jet engine and the results of the centre of gravity

----

Position of the jet engine on the aircraft : mass and gravity centre of the single jet engine

----

1. Declaration
2. Information on module
3. Check if a data protocol is defined
4. Constant
5. Data bank used
6. Description of the data set fixed
7. Reading the necessary data from the data bank
8. Monitoring the inputs
9. Computation part

9.1 Position of the jet engine on the aircraft

10. Display the results

As outlined in orange, most of those subroutines call then directly general subroutines to compute the results, for example, for M**D6A**: **TWHP1,** or **TWHP2** .

MD14 focuses on the performance of the jet engine

MD14 does the calculations concerning the necessary thrust of each jet engine in specified cases.

The case considered is:

- Thrust needed during take off.
- Thrust needed for the beginning of the cruise
- Thrust needed at the end of the cruise
- Thrust needed at approach flight
- Thrust needed during an interrupted landing

MD33 is more oriented on the performance characteristics of the jet engines. It computes the

-Maximal thrust

-Specific Fuel Consumption

- Thrust moment (Yaw, Roll und Pitch) (chocked, all jet engines working, reference point: nose)

- Thrust angle/Nicken (chocked, all jet engines working)

-NOx-Emissions characteristics

- CO-Emissions

- HC-Emissions

Algorithm:

1. Declaration
2. Information on module
3. Data bank used
4. Read the necessary data from the data base
5. Calculation of the jet engine performance characteristics

   -jet engine model is known: CALL VTW1

   -jet engine model = HEINZE/DECHOW (1991), jet engine configuration = HPC-HPT

   CALL TWKEN2 (IAUS(IMODUL),1)

   -jet engine model = HEINZE/DECHOW (1991),

   jet engine configuration =FAN-HPC-HPT-LPT,

   CALL TWKEN2 (IAUS(IMODUL),2)

   -jet engine model = HEINZE/LEITNER (1999),

   jet engine configuration =FAN-LPC-HPC-HPT-LPT,

   CALL TWKEN2 (IAUS(IMODUL),10)

   -jet engine model = MATTINGLY/HEINZE(2006),

   jet engine configuration =FAN-LPC-HPC-HPT-LPT,

   CALL TWKEN2 (IAUS(IMODUL),11)

   Jet engine model is unknown : send an error flag.

5.2 Jet Engines Libraries

There are seven pre-existent jet engine libraries: TW1_LBY, TW2_LBY, TW3_LBY, TW4_LBY, TW5_LBY, TW6_LBY and GEO_TW1.

TW1_LBY: This jet engine library contains the subroutines computing most of the things related to the jet engines in PrADO. It contains 30 subroutines.

Tdat is an interface for jet engine simulation and it computes requested data

1. Declaration
2. Information on module
3. Reading necessary data from the data bank
4. Choice of the computation
    4.1. Computation for an ersatz jet engine
        - Thrust CALL TDAT001
        - SFC CALL TDAT002
        - Thrust moment CALL TDAT003
        - Gas emission CALL TDAT004
        - Thrust moment (pitch, yaw, roll) CALL TDAT005
    4.2. Computation for a given jet engine and a given kind of fuel
        - Thrust CALL TDAT101
        - SFC CALL TDAT102
        - Gas emission CALL TDAT103
    4.3. Computation for a chosen jet engine and given kind of fuel
        - Thrust CALL TDAT201
        - SFC CALL TDAT202
        - Gas emission CALL TDAT203
        - Thrust angle CALL TDAT204
        - Thrust moment CALL TDAT205
    4.4 Computation for a chosen jet engine and a flight phase
        - Thrust CALL TDAT301
        - SFC CALL TDAT302
        - Gas emission CALL TDAT303
        - Thrust angle CALL TDAT304
        - Thrust moment CALL TDAT305
    4.5. Computation of general values for a propulsion system and a defined flight phase
        - Thrust CALL TDAT401
        - Temporary Specific Fuel consumption CALL TDAT402
        - Temporary Gas emission CALL TDAT403
        - Thrust moment CALL TDAT404

4.99. Help function
Estimation of the critical jet engine in operational conditions
CALL TDAT1001
4.100 Computation mode is unknown (send error)

It would be useless to detail the structure of each subroutine called in tdat since they all have the same one:

1. Declaration
2. Information on module
3. Check if a data protocol is defined
4. Leading computation
   Setting initial values
   Saving inputs
   Calling data from the data bank
   Loop on the jet engine
      - Determination of the jet engine type (TWPH1(3…
      - Determination of the fuel for the chosen jet engine type
      - Call jet engine model and computes researched data
      If model = TL2 CALL TL2
      If model = ZTL2 CALL ZTL2
      If model = ZTL10 CALL ZTL10
      If model = ZTL11 CALL ZTL11
      If model = ZTL12 CALL ZTL12

Saving data

TWHP1 and **TWHP2** (TriebWerk HilfsProgram) are programs aiding to deal with information about the kind of jet engine and the kind of fuel which is used.
Depending on the first parameter given to the TWHP subroutine you can compute many different things and this program is used in different points of the program (MD6A and TL2G)

So **TWHP1** can perform 5 different tasks, and therefore its input and output are different according to the task accomplished:

1 Read the number of the jet engine types	
Input :none	Output: Number of the jet engine
2-give a designation for every jet engine	
Input : None	Output: Designation of the jet engine type 1 Designation of the jet engine type 2 …

	Designation of the jet engine type n
3-give the jet engine type of a specified jet engine	
Input : Number of the jet engine	Output: Designation of the jet engine type
4-give the jet engine identifier.	
Input : None	Output: 1. Identifier of the jet engine type
5-give the operating conditions of a specified jet engine in a specified flight case.	
Input : 1. Number of the jet engine 2. Designation of the flight phase	Output: 1. Designation of the jet engine type

Table 5.1 Functions of the TWHP1 subroutine

TWHP2 performs 8 different tasks with different inputs and outputs:

1-Give the kind of fuel used for the conception of the airplanes	
Input : Designation of jet engine type	Output: Kind of fuel used for this draft case
2-give the number of the fuel used for a specified jet engine	
Input : Designation of jet engine type	Output: Designation of the jet engine type
3-Indices of every kind of fuel used for a specified aircraft	
Input : Designation of jet engine type	Output: Designation of fuel type 1 Designation of fuel type 2 … Designation of fuel type n
4-give density of the fuel chosen	
Input : Designation of fuel type	Output: Density
5-give the specific calorific value of a chosen fuel	
Input : Designation of fuel type	Output: Specific calorific value
6-give the adiabatic exponent and the heat capacity of the air incoming	
Input : Designation of fuel type	Output: Adiabatic value (air) Heat capacity (air)
7-give the adiabatic exponent and the heat capacity of the exhaust air	
Input : Designation of fuel type	Output: Adiabatic value (exiting air)

	Heat capacity (exiting air)
8-give the fuel used in a specified flight case for a specified jet engine	
Input : Number of jet engine Designation of flight case	Output: Designation of the fuel type

Table 5.2 Descriptions of the TWHP2 subroutines

TWKEN2 (for TriebWerkKENnfelds)

This subroutines compute jet engine performance characteristics:

- Maximal thrust
- Specific Fuel consumption (with throttling)
- NOx-Emissions
- CO-Emissions
- HC-Emissions
- H2O-Emissions

Its algorithm is as follows:

1. Declaration
2. Information on module
3. Check if a data protocol is defined
4. Constant
5. Data bank used
6. Description of the data set
7. Display results
8. Calculation of the jet engine performance characteristics
   8.0. Erase data in the data bank
   8.1. Read data from the data bank
   8.2. Define support for the performance characteristic
   8.3. Compute the performance characteristics and save them in a file
   8.4 Data set of the jet engine performance characteristics and save it in the PrADO data bank
   8.99 redefine the control parameter
9. 9. Display results

TW2_LBY: This contains 6 subroutines. Most of them are computation models for different types of jet engines or more precise models.

- **TL2** computes for a TL- jet engine the temporary thrust, momentary thrust, the throttle degree, consumption and the necessary static thrust in all flight attitudes with a simple approximation method considering a cyclic thermodynamic process.

Algorithm:

Declaration
Initializing output value

Data bank used
Checking the input
Reading values from the data bank
- Specific calorific value of the fuel
- Static thrust
- Maximal turbine entry temperature
- Global compressor ratio in take off phase
- Isentropic ratio in the entry
- Isentropic ratio in the compressor
- Isentropic ratio in the turbine
- Isentropic ratio in the nozzle
- Ignition capacity of the fuel
Display the input
-Presentation of the data
-Computation of the throttle degree
-Air data in case of ground proximity
-Atmosphere data in considered flight case
-Computing thrust ratio
Computing G
Computing G0
Computing the temporary thrust to Static thrust
Computing the temporary thrust STW
Computing the specific fuel consumption
Iteration of the mass flow ratio until 0.1%
Display the outputs (debriefing)

TL2G (Jet engine Geometry)

This subroutine computes geometrical Dimensions for a TL – jet engine

Algorithm:

1. Declaration
2. Information on module
3. Check if a data protocol is defined
4. Constant
5. Data bank used
6. Description of the data set fixed
7. Reading necessary data from the data base
- Static thrust
- bypass ratio
Name of the file with the jet engine lines
- Jet engine identifier
- Masse correction factor

8. Output control (beginning)
9. Computation part
    - Maximal jet engine diameter
    - Maximal jet engine length
    - Data set for the presentation of the jet engine contour
        - Initial value
        - Opening reading canal
        - Building jet engines line
        - Closing reading canal
        - Saving in the data bank
    - Jet engine mass
    - Gravity centre position (Hypotheses: gravity centre of the jet engine = gravity centre of the jet engine volume)
100. Output control (End)

VTW1
This subroutine reads data of a given jet engine
Algorithm:

1. Declaration
2. Information on module
3. Check if a data protocol is defined
4. Constant
5. Data bank used
6. Description of the computed data set used
7. Reading geometrical data, mass and gravity centre
    - Reading needed data
        -Name of the file with the jet engine data
        -Jet engine identifier
        - Weight Correction Factor

ZTL2 Computes for a twin spool jet engine (Zweistrom Triebwerk Leistungen) the temporary thrust, momentary thrust, the throttle degree, consumption and the necessary static thrust in all flight attitudes with a simple approximation method considering a cyclic thermodynamic process.
Algorithm:

Declaration
Initialization of result variable
Databank used
Checking the entry
    - control parameter concerning the jet engine type
    - control parameter concerning the fuel type

Reading data from the data bank

Lower fuel heat value CALL TWHP2
jet engine static thrust
Maximal turbine entry temperature
Bypass ratio
General compressor ratio of the compressor during take off case
Isentropic efficiency of the entry
Isentropic efficiency of the fan during the take off case
Isentropic efficiency of the fan during the cruise flight case
Isentropic efficiency of the compressor
Isentropic efficiency of the nozzle
Ignition capacity of the fuel

Output control-INPUT
Presentation of the data
- Computation of the throttle grad
- Air properties in ground properties
- Atmosphere data

ZTL2G

This subroutine computes geometrical Dimensions for a twin spool jet engine
Algorithm:

1. Declaration
2. Information on module
3. Check if a data protocol is defined
4. Constant
5. Data bank used
6. Description of the computed data
7. Reading necessary data
   - Static thrust
   - Bypass ratio
   - name of the file with specification of the jet engine lines
   - Identifier of the jet engine type CALL TWHP(4, ...
   - Weight Correction Factor/Propulsion CALL HWE2
8. Output control (Initialization)
9. Computation part
   - Maximal jet engine diameter
   - Maximal Fan diameter
   - Maximal gas generator diameter
   - Maximal jet engine length
   - Length of the fan part
   - Length of the gas generator part

- data set for the presentation of the jet engine contour
    - Value for initialization
    - Opening reading canal
    - build jet engine lines
        - Twin spool jet engine
        - Fan
        - Gas generator
        - TL jet engine
        - Complete jet engine
    - Opening reading canal
    - Saving data
    - Jet engine weight
    - Gravity centre

100. Displaying results values

TW3_LBY: is composed of 19 subroutines, called in twken subroutines. They serve geometrical purpose as well as performance computation .The ztl11 and ztl12 series are more like jet engine's performance model similar to the TW2_LBY.

TWH1:
This subroutine helps to compute flow data.
It has nine computation modus:
The three first modus concern the general temperature, pressure, density with the static temperature and the mach number as input and the 4^{th},5^{th} and 6^{th} modus gives the general value with the static one given.
The 7^{th} gives the general density with the general pressure and temperature given.
The 8^{th} computes the MFP parameter with the flow mach given and the 9^{th} does the inverse.

TWH2A: produce a matrix describing geometrically the jet engine contour using the number of the cut, the x-coordinate of the first and second cut, the diameter of the first and second cut.

TWH2B: produces a matrix describing geometrically the jet engine contour using a lot of very detailed input. It is advisable to go in the subroutine to see everything needed.

TWH3A produces a matrix describing geometrically the nacelle using the max IWERT(1) maximal nacelle length, the maximal jet engine diameter, maximal density of the nacelle and a dimensionless description of the nacelle cut characteristics.

TWH3B does the same as twh3a but it takes the contour from a template files.

The TWK series:

There are seven subroutines beginning with “twk”. Each is concerned with a particular jet engine part and each has different modus permitting it to compute general variables.

Twk1 : Entry
1. Computation of the general temperature at the end of the entry
2. Computation of the general pressure at the end of the entry
3. Computation of the general density at the end of the entry
4. Front surface of the flow channel
5. External diameter
6. Length
7. Computation of the flow mach (MFP parameter is given)
8. Computation of the front surface of the entry and computation of the front surface of the undisturbed flow from the atmosphere data
Twk2 : Compressors (Fan, low pressure compressor, high pressure compressor)
1. Computation of the general temperature at the end of the compressor
2. Computation of the general pressure at the end of the compressor
3. Computation of the general density at the end of the compressor
4. Computation of the number of the compressor’s stage and of the efficiency pressure ratio per compressor stage
5. Computation of the compressor ratio
6. Front surface of the flow channel
7. External diameter
8. Length
9. Weight
10. Weight of the holding frame
11. Computation of the general temperature at the end of the compressor (with the polytropic ratio)
12. Computation of the isentropic efficiency
13. Computation of the pressure ratio
14. Computation of the temperature ratio
15. Front surface of the flow channel (with the external diameter and the bypass ratio given)
16. Computation of the flow mach number (MFP parameter is given)
17. Computation of the temperature ratio (with the pressure ratio given)
Twk3 : Junction part
1. Computation of the general temperature at the end of the junction part
2. Computation of the general pressure at the end of the junction part
3. Computation of the general density at the end of the junction part
4. mach number of the flow

Twk4 : Convergent thrust nozzle
1. Computation of the critical pressure ratio in nozzle entry
2. Computation of the general pressure in the nozzle entry
3. Computation of the static pressure in the nozzle entry
4. Computation of the mach number in the nozzle entry
5. Computation of the speed in the nozzle entry
6. Computation of the static temperature in the nozzle entry
7. Computation of the flow mach number
8. Front surface of the flow channel
Twk5 : Combustion chamber
1. Computation of the general pressure at the end of the combustion chamber
2. Computation of the general density at the end of the of the combustion chamber
3. Computation of fuel mass flow
4. Length
5. Weight
6. Computation of the mach flow with MFP parameter given
7. Front surface of the flow channel
Twk6 : Low and high pressure turbine
1. Computation of the general temperature at the end of the turbine
2. Computation of the general pressure at the end of the turbine
3. Computation of the general density at the end of the turbine
4. Computation of the number of the turbine’s stage and of the efficiency pressure ratio per turbine stage
5. Front surface of the flow channel
6. External diameter
7. Length
8. Weigh
9. Weight of the holding frames
10. Computation of the general pressure at the end of the turbine(with the polytropic ratio given)
11. Computation of the isentropic efficiency
12. Computation of the pressure ratio
13. Computation of the turbine performance
14. Computation of the flow mach number (MFP parameter is given)
15. Computation of the Temperature ratio
16. Computation of the maximal RPM (with the maximal peripheral velocity on the rotor

tip)
17. Computation of the maximal RPM (with the maximal tension in rotor given)
Twk7 : Mixer
1. Computation of the general pressure at the end of the combustion chamber
2. Computation of the general density at the end of the of the combustion chamber
3. Computation of fuel mass flow
4. Length
5. Weight
6. Computation of the flow mach number with MFP parameter given
7. Front surface of the flow channel

Table 5.3 Functions of the TWK… surboutines

As already said, ZTL subroutines from the TW3 library are more like computation model.

ZTL11

This subroutine computes for a twin spool jet engine the thrust, the throttle degree and the needs for a given flight point with a standard fuel consumption under consideration of a thermodynamic cycle process. The geometry, the weight and the gravity centre of the jet engine is determined afterward.

Algorithm:

Declaration
No Go criteria
Initializing the output values
Data bank used
Reading the data from the data bank
    Static thrust
    Static thrust from the jet engine design
    Flight mach with the jet engine design point
    Flight altitude for the jet engine design point
    Thrust for the jet engine design point
    Checking the computation modus and pass
    Display results
Design of the jet engine
    Computation of the maximal turbine temperature entry
    Design of the jet engine CALL ZTL11D
    Computation of the jet engine thrust for the design point CALL ZTL11OFF
    Computation of the jet engine thrust at the ground CALL ZTL11OFF
    Scaling the thrust to the design point
    Checking the variation
    Saving value of the static thrust at the design point

Computing the jet engine performance data
    Constructing the value of the maximal jet engine entry temperature
    Display results

ZTL11D
This subroutine leads the design of the jet engine for the following configuration:
FAN-LPT- HPT-CC-HPT-LPT
Hypothese:
- Definition of the cyclic process
- Computation of the jet engine geometry
- Computation of the jet engine's mass and gravity centre

Input value
Control parameter
Design point of the jet engine
Air and fuel properties
Jet engine data
Entry data
Fan data
Junction data (stage 2)
Nozzle data (stage 2)
Low-pressure compressor data
High-pressure compressor data
Combustion chamber data
High-pressure turbine data
Data of the second mixer
Low-pressure turbine data
Junction data (stage 1)
Plug data
Nozzle data (stage 1)
Jet engine stage data
Mass and gravity centre
Output value

Declaration
Definition of the data bank used
Reading data from the data bank
Computation of the missing data
Monitoring the input
Computation model
    Plane 0: undisturbed flow
        Air properties

Mach number
Velocity
Mass flow
General temperature CALL TWH1 (1,…)
General pressure CALL TWH1 (2,…)

Plane 1: flow in the jet engine entry
Mass flow
Mach number
General temperature
General pressure
Surface CALL TWK1(8…)
External diameter CALL TWK1 (5…)

Plane 2: flow between the entry and the FAN
Mach number
Mach flow
General temperature CALL TWK1(1,…)
General pressure CALL TWK1(2,…)
Surface CALL TWK2(2,…)
External Diameter CALL TWK2(7,…)

Plane 3: flow behind the fan
Mass flow
General temperature CALL TWK2(11,…)
General pressure CALL TWK2(2, …)
Isentropic efficiency CALL TWK2 (12,…)
Fan performance CALL TWK2(5,…)
Temperature ratio
External diameter
Surface
Number of the compressor stage wit its compressing ratio
Mach number

Plane 7s: flow behind the junction and before the nozzle
Mass flow
General temperature CALL TWK3(1,..;)
General pressure CALLTWK3(2,…)

Plane 9s : Flow behind the nozzle II
Mass flow
General temperature
Critical pressure ratio CALL TWK4(1,…)
General pressure CALL TWK4(2,…)
Static pressure at the nozzle end CALL TWK4(3,…)
Static temperature at the nozzle end CALL TWK4(6,…)
Mach number at the nozzle exit CALL TWK4(4,…)

Velocity at the nozzle exit CALL TWK4(5,…)
Surface CALL TWK4(8,…)

(As it can be seen the structure is very detailed, and as it is relatively similar for the next plane, if more information is needed, one should give a look at the file itself)

Plane 3 ‘ ‘ V , flow before the LPC
Plane 3 ‘ ‘ Flow between the LPC and the HPC
Plane 3 : flow before the HPC
Plane 3: Flow behind the HPC
Plane 3a : flow before the combustion chamber
Plane 4 flow behind the combustion chamber
Plane 4a Flow behind the first mixer and before the HPT
Plane 4b flow behind the HPT and before the second mixer
Plane 4c flow behind the second mixer and before the LPT
Plane 5 flow behind the LPT and the second junction
Plane 7 flow behind the junction and before the nozzle
Plane 9 flow behind the nozzle 1
MFP parameter in the nozzle exit
Performance data
Iteration on the mass flow to determine the needed thrust
RPM limitation from the maximal speed at the blade tip
RPM limitation due to the centrifugal force
Jet engine Entry component
Jet engine comoment FAN
Jet engine comoment Low pressure turbine
Junction between LPC and HPC
Jet engine comoment High pressure turbine
Jet engine comoment Combustion chamber
Junction between the combustion chamber and the HPT
Jet engine comoment HPT
Junction between the HPT and HPT
Jet engine comoment low pressure turbine
Jet engine component junction part / mixer
Plug
High Pressure stage
Low Presssure stage
Jet engine
Geometrical description
(Description of each jet engine parts)
Display the results

ZTL11OFF has almost the same structure for the same objectives.

The ZTL12 subroutines (ZTL12, ZTL12D and ZTL12OFF) follows the same schemes as ZTL11 one.
Those files are definitely the ones which would be interesting to copy and change for someone who would like to add a new jet engine and would be interested in giving very detailed results pages on the jet engine design.

TW4_LBY: is empty (The small info text in the folder indicates that it should be similar to TW3_LBY).
TW5_LBY: this library provides 3 subroutines for CO, NoX, Hc emission
TW6_LBY: this library provides 19 subroutines for nacelle geometry computation.
GEOTW1_LBY: this library provides 8 subroutines for the jet engine geometry computation.
The main subroutine is called **geotw** and it has a lot of sub options which call other subroutines (similarly to **tdat** in TW1_LBY)

5.3 Jet Engine Data Bank and Templates

5.3.1 Data Banks

Here follows a small description of the jet engine's data bank (DB 8). There are 145 variables saved.

Variable	Unit	Description
IMD14P1		Control parameter/Estimation of the thrust needed while flying in the critical altitude (1 Jet engine out)
IMD14P2		Control parameter/Estimation of the thrust needed at the beginning of the cruise flight (Flight with maximal payload)
IMD14P3		Control parameter/Estimation of the thrust needed at the end of the cruise flight (Flight with max. Payload)
IMD14P4		Control parameter/Estimation of the thrust needed at the end of the cruise flight (Flight with max. fuel)
IMD14P5		Control parameter/Estimation of the thrust needed in landing flight (1 Jet engine out)
IMD14P6		Control parameter/Estimation of the thrust needed by a missed approach during landing
IMD14P7		Control parameter/Estimation of the thrust needed for the respect of the start track length
IMD14P8		Control parameter/Estimation of the thrust needed at the beginning of the cruise flight (Flight with max. payload /1 jet engine out)
IMD14P9		Control parameter/Estimation of the thrust needed at the beginning of the cruise flight (Flight with max. payload /1 jet engine out)

ISTW		Control parameter/Method/Jet engine performance
IPTW1		Control parameter (1:GummiTW,2:fixed Jet engine)
IPTW2		Control parameter/ Jet engine type
DATITW1		File Name/Jet engine geometry
DATITW2		File Name/Jet engine nacelle geometry
DATITW3		File Name/Jet engine pylon geometry
DATITW4		File Name/Data for predefined jet engine
NTW		Number of jet engines
SOTW	N	Jet engine static thrust (Ma=0,H=0 km)
BYPASS		Bypass ratio
T3GMAX	K	Maximale Turbin Entry Temperatur
DELTAT3G	K	Temperatur decrease in cruise flight
PIOATO		General pressure ratio (Start)
VHDVNDVTO		General pressure ratio in the HPC/LPC(Start)
PIOACR		General pressure ratio (cruise flight)
PIOACHAR		Caracteristic value of the general pressure ratio
PIEINL		Pressure ratio in the entry
MAEINL		Mach number in the entry plane
LAMBDA1	Grad	Effective diffusor angle
SPINNER	Grad	Spinner angle
PIFANTO		Fan pressure ratio(Start)
NUEISFAN		Maximal isentropic compressor efficiency/FAN
NUEFANTO		Isentropic compressor efficiency /FAN (Start)
PIUEII		Pressure ratio/Jonction of the FAN nozzle
MAFANEIN		Mach number /FAN entry
MAFANAUS		Mach number/FAN exit
HTFANEIN		Hub-shroud diameter ratio/FAN entry
HTFANAUS		Hub-shroud diameter ratio, FAN exit
LVORFAN	m	Forward FAN Frame Length (LEading edge(VK) FAN FrameVK FAN)
UTFANMAX	m/s	Maximale FAN TIP celerity
NFANROT		Mean blade number of the FAN Rotor
NFANSTA		Mean blade number of the FAN Stator
ARFANROT		Rotor heigth/Rotor width FAN (mean)
ARFANSTA		Stator heigth/Stator width FAN (mean)
CVFANROT		Constant of volume per FAN blades (rotor)
CVFANSTA		Constant of volume per FAN blades (stator)
RHOFANROT	kg/m**3	Density of the FAN rotor material
RHOFANSTA	kg/m**3	Density of the FAN stator material
NUEISNDV		Maximal isentropic efficiency /LPC
NUENDVTO		Isentropic compressor efficiency /NDV (take off)
MANDVEIN		Mach number/ LPC entry
MANDVAUS		Mach number/LPC exit
PINDVST		Compressor ratio per LPC stages
HTNDVEIN		Hub-shroud diameter ratio/LPC entry
HTNDVAUS		Hub-shroud diameter ratio/LPC exit
LUEBER1	m	Length of the jonction between LPC and HPC
NNDVROT		Mean blade number of the LPC rotor

NNDVSTA		Mean blade number of the LPC stator
ARNDVROT		Rotor heigth/Rotor width LPC (mean)
ARNDVSTA		Stator heigth/ Stator width LPC(mean)
CVNDVROT		Constant of volume per LPC blade (Rotor)
CVNDVSTA		Constant of volume per LPC blade (stator)
RHONDVROT	kg/m**3	Density of the LPC rotor material
RHONDVSTA	kg/m**3	Density of the LPC stator material
NUEISHDV		Maximal isentropic compressing efficiency /HPC
NUEHDVTO		Isentropic compressing efficiency/HPC (take off)
MAHDVEIN		Mach number /HPC entry
MAHDVAUS		Mach number /HPC exit
PIHDVST		Compressing ratio per HPC stage
HTHDVEIN		Hub-shroud diameter ratio/HPC entry
HTHDVAUS		Hub-shroud diameter ratio/HPC exit
NHDVROT		Mean blade number of the HPC rotor
NHDVSTA		Mean blade number of the HPC stator
ARHDVROT		Rotor heigth/Rotor width HPC (mean)
ARHDVSTA		Stator heigth/Stator width HPC (mean)
CVHDVROT		Constant of volume of the HPC blades (Rotor)
CVHDVSTA		Constant of volume of the HPC stator
RHOHDVROT	kg/m**3	Density of the HPC rotor material
RHOHDVSTA	kg/m**3	Density of the HPC stator material
PIBK		Compressing ratio of the combustion chamber
NUEA		Heat efficiency
VBK	m/s	Mean celerity in the combustion chamber
TBK	s	Retention time of the gas part of the combustion chamber
RHOBK	kg/m**3	Density of the combustion chamber material
SIGMABK	N/mm**2	Maximal tensil stress in combustion chamber wall
NUEISHDT		Maximal isentropic Turbine Efficiency/HPT
NUEHDTTO		Isentropic Turbine efficiency/HPT (take off)
MAHDTEIN		Mach number /HPT entry
MAHDTAUS		Mach number /HPT exit
PIHDTST		Compressing ratio per HPT stages
HTHDTEIN		Hub-shroud diameter ratio/ HPT entry
HTHDTAUS		Hub-shroud diameter ratio/HPT entry
LUEBER3	m	Length of the jonction between HPT and LPT
NHDTROT		Mean number of blades of the HPT rotor
NHDTSTA		Mean number of blades of the HPT stator
ARHDTROT		Rotor heigth/ Rotor width HPT (mean)
ARHDTSTA		Stator heigth/Stator width HDT (mean)
CVHDTROT		Volume constant of the HPT blades (Rotor)
CVHDTSTA		Volume constant of the HPT Stator
RHOHDTROT	kg/m**3	Density of the HPT rotor material
RHOHDTSTA	kg/m**3	Density of the HPT stator material
SIGMAHDT	N/mm**2	Maximal tensile stress in the HPT blades (rotor)
PIUEI		Pressure ratio /Jonction LPT/nozzle
NUEISNDT		Maximal isentropic Turbine efficiency/LPT
NUENDTTO		Isentropic Turbine efficiency/LPT(take off)

MANDTEIN		Mach number/ LPT entry
MANDTAUS		Mach number/LPT exit
PINDTST		Pressure ratio per LPT stage
HTNDTEIN		Hub-shroud diameter ratio/LPT entryt
HTNDTAUS		Hub-shroud diameter ratio/ LPT exit
NNDTROT		Mean number of blades for the LPT rotor
NNDTSTA		Mean number of blades for the LPT stator
ARNDTROT		Rotor height/Rotor width LPT(mean)
ARNDTSTA		Stator height/ Stator width LPT(mean)
CVNDTROT		Constant of volume for LPT blade
CVNDTSTA		Constant of volume for LPT Stator
RHONDTROT	kg/m**3	Density of the LPT rotor material
RHONDTSTA	kg/m**3	Density of the LPT stator material
SIGMANDT	N/mm**2	Maximal tensile stress in the LPT rotor
LMIXER	m	Length of the mixer
LZUSATZ	m	Additional length of a jonction LPT / mixer
LAMBDA2	Grad	Angle of the jonction cone
PHISD		Resultant celerity factor of the thrust nozzle
NUEMW1		Mechanical efficiency/LP shaft
NUEMW2		Mechanical efficiency/HP shaft
RHONDW	kg/m**3	Density of the LP shaft material
RHOHDW	kg/m**3	Density of the HP shaft material
TAUNDW	N/mm**2	Maximal tranverse tension in the LP shaft
TAUHDW	N/mm**2	Maximal tranverse tension in the HP shaft
TMITNDW	m	Mean density of the LP shaft
TMITHDW	m	Mean density of the HP shaft
NTG		Number of jet engine nacelle
TGPOS1		Vector Positon/1.jet engine nacelle
TGPOS2		Vector Positon/2.jet engine nacelle
SRTG	mm	Harshness of the surface /jet engine nacelle
DITG	m	Maximal nacelle density
FLUEBER2		nacelle extension/ jonction diffusor (0 ... 1.3)
SPTG1		Dimensionless Geometrical description/process of the nacelle cut(nacelle in one part)
FGONDEL	kg/m**2	Weight per surface of the nacelle
SRTP	mm	Harshness of the surface/turbojet pylon
DITP	m	Maximal Pylon density
ZHIATPF		X coordinate of the forward pylon attach on the wing (X/TF)/reference point : wing nose in local cut
ZHIETPF		X coordinate of the aft pylon attach(X/TF)/reference point: flight nose in local cut
ZHIATPR		X coordinate of the forward pylon attach on the fuselage (X/LGR)/reference point : fuselage nose
ZHIETPR		X coordinate of the aft pylon attach on the fuselage (X/LGR)/ reference point : fuselage nose
OMEGATPR	Grad	Angle of the pylon attach on the fuselage

Table 5.4 List of the variables of the Data bank 8

5.3.2 Templates Files

In the template folder, there are two folders concerning directly the jet engines:

The ANTRIEBSANLAGE folder has four files describing specific jet engines:

<table>
<tr><th>File name</th><th>Content</th></tr>
<tr><td>CFM56-5A4.DAT</td><td rowspan="2">Global Jet engine data
Jet engines data
Nacelle data
Jet engine pylon data
Thrust values
SFC value</td></tr>
<tr><td>GECF6-80E1.DAT</td></tr>
<tr><td>V_CFM56-5A4.DAT</td><td rowspan="2">Global Jet engine data
Jet engines data
Thrust values
SFC value
NOx-Emission values
CO-Emission values
HC-Emission values
H2O-Emission values</td></tr>
<tr><td>V_GECF6-80E1.DAT</td></tr>
</table>

Table 5.5 Files of the ANTRIEBSANLAGE folder

The TRIEBWERKS folder contains four data files:

File name	Description
TG1.DAT	This files contains a dimensionless geometrical description for a nacelle (in one part)
TG2.DAT	This files contains a dimensionless geometrical description for a nacelle (in two parts)
TL1.DAT	This files contains a dimensionless geometrical description for a single spool jet engine
TPP1.DAT	This files contains a dimensionless geometrical description for a jet engine pylon
TW1.DAT	This files contains a dimensionless geometrical description for a nacelle (in one part) for a turbofan
TW2.DAT	Same as TW1.DAT
ZTL1.DAT	This files contains a dimensionless geometrical description for a twin spool jet engine

Table 5.6 File of the TRIEBWERK folder

6 Methodology and Description of the GUI

6.1 Methodology

The methodology to describe is the process which permits to understand a subroutine or the program called by a certain button and how to start to find any information inside the program source code.
The first step is to find out which program is called. This can be done by looking at the monitor of the Java interface: it displays clearly the name of the subroutine or of the script called. A simple window research (Ctrl+ F) in the PrADO folder allows us to find the file researched.

In the case of a Fortran file describing a subroutine, the code is supposed to be self descriptive which means they almost always include the name of the author, the date of creation and update of the file, a short description at their beginning as well as a listing of the in- and output variables . The structure of the files is often also written in the files.
This implies the descriptions and the structure shown in this report are the ones provided by the author of the subroutine, translated in English.

What describes the interconnection between the libraries is the CALL of the subroutines. I have chosen not to be consistent in the description of the calls in order to show only the most important ones.

6.2 Description of the GUI

The GUI process in PrADO is under copyright of the IFL, so how to reproduce it will not be described here but how to analyze it to know what it exactly calls can be shown.

During this research, a free java compiler, Eclipse, was installed to open the java files since they’re not adapted to be opened by the window notepad. After a quick analysis, it can be easily seen that the java programs don’t describe the PrADO windows but only provide the basic code which reads script files and produces the windows from it. So in order to know which program is called by which button, the scripts have to be studied.
They are located in PRADO/GUI/… and can be opened by the windows text editor. The most important one is the script of the main window:

A shortened version of this file is presented here:

```
/*********************************************************************
 **  Global Functions                                        **
 *********************************************************************/
GLOBAL  ("PrADO/TA2") {
METHODS () {
BUTTON  ("PrADO Projektdefinition")       {EXECUTE ("javaw -cp .\SOG sog
.\GUI\SCRIPT_COMMON\PrADO-PROJEKT-D-GUI.scr");}
 […]
/*********************************************************************
 ** Program functions of PrADO                               **
 *********************************************************************/
LABEL   ("Hilfen");
BUTTON  ("Vorgabedaten einlesen")         {EXECUTE (".\BAT\prado1.bat");}
BUTTON  ("Sicherheitskopie erstellen")    {EXECUTE (".\BAT\prado2.bat");}
BUTTON  ("DMS Datenbankeditor")           {EXECUTE ("javaw -cp .\SOG sog
.\GUI\SCRIPT_COMMON\PrADO-DMSEDITOR-D-GUI.scr");}
[…]
LABEL   ("Darstellung/TECPLOT");
BUTTON  ("Historie/Entwurfsanalyse")      {EXECUTE (".\BAT\DispTec-GEIT.bat");}
BUTTON  ("Historie/Parametervariation")   {EXECUTE (".\BAT\DispTec-
GPVAR_B.bat");}
BUTTON  ("Historie/Optimierung")          {EXECUTE (".\BAT\DispTec-GOVAR_B.bat");}
BUTTON  ("Diagramme")                     {EXECUTE ("javaw -cp .\SOG sog
.\GUI\SCRIPT_JAVA\PrADO-Grafik\2dkurvea-gui.scr");}
[…]
LABEL   ("Gesamtentwurfsprogramm");
BUTTON  ("Einzelanalyse (EA)")            {EXECUTE (".\BAT\prado10.bat");}
BUTTON  ("Parametervariation (PV)")       {EXECUTE (".\BAT\prado11.bat");}
BUTTON  ("Optimierung (OP)")              {EXECUTE (".\BAT\prado12.bat");}

LABEL   ("Einzelmodule");
BUTTON  ("Anforderungen (MD1)")           {EXECUTE (".\BAT\md1.bat");}
BUTTON  ("Rumpfgeometrie (MD2)")          {EXECUTE (".\BAT\md2.bat");}
 […]
LABEL   ("Datenbearbeitung")
BUTTON  ("DB1 Programmsteuerung")         {EXECUTE ("javaw -cp .\SOG sog
.\GUI\SCRIPT_JAVA\PrADO-TA2\DB1\IO-PrADO-TA2-DB1-GUI.scr");}
BUTTON  ("DB2 Anforderungen")             {EXECUTE ("javaw -cp .\SOG sog
.\GUI\SCRIPT_JAVA\PrADO-TA2\DB2\IO-PrADO-TA2-DB2-GUI.scr");}
[…]
```

Note: this symbol "[…]" means a part of the code is intentionally not shown

Looking at this file shows that there are two possibilities for a button: it executes either a batch file, or it executes the java shell (javaw -cp .\SOG sog) and a script at the same time.

The first option calls a batch file which launches the executable having the same name. The executables are compiled from source files which include a description of the files.

The second calls another java interface which can be analyzed by the same process as the main windows.

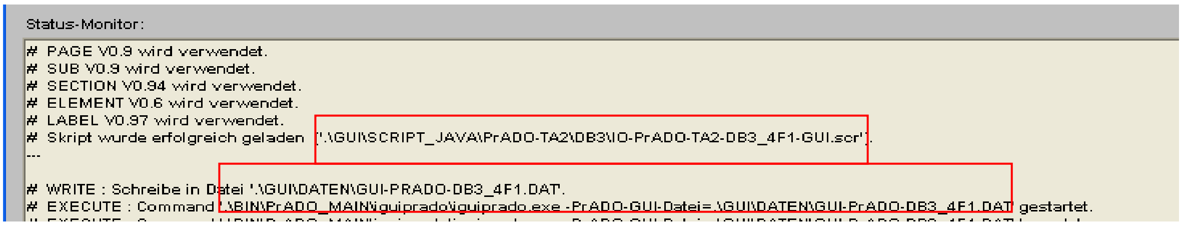

Figure 6.1 Detail of the status monitor

As shown in figure 6.1, looking at the executable launched by the main windows in the status monitor is also interesting since it shows which subroutines and files are implied by the button of the window. After a quick research, it can be found that they're located in the PrADO\SOURCE\MAIN\ folder and functions which are not modules, like the project definition, the data bank generation or the safety copy are performed by subroutines of this folder called "prado1" to "prado12". Tecplot related buttons have other names which can be found in the files.

Using this method should help to understand any part of PrADO launched by the main window.

7 Discussions

Many things make this report incomplete: as a program, description files in it should be analyzed but PrADO involves at least three different software and 1600 files.
Besides, those files are constantly evolving so it would be impossible to make a definitive description of the PrADO structure.
The following discussion tends to gives solutions to these issues.

7.1 Language Discussion

Fortran is the language of most of the PrADO structure so it is a big characteristic of the global program. Not only because of its capacities in term of simplicity, safety and velocity but it also determines the possibilities of evolution and the graphical interfacing.

In fact, Fortran is not evolving much and the latest possibilities provided by other language such as Java or C++ are missing when one works on such a complex program.
Of course, translating the program into another language would be a huge work and having those possibilities might not be enough to take such a decision, but if a discussion should be led on the subject, then the following arguments could be interesting.

Clearly, what would be useful is:

-having a free compiler downloadable for every person working on the project. (Java has Eclipses, C++ has Visual Studio 2005 express). Those free compilers keep evolving all the time, providing new fastening techniques, which Microsoft Compaq Fortran visual studio does not.

- having access to free standard libraries which permit to do different things with one language: graphical interfacing, graphical output or data management. All this would allow us to have a more simplified program structure with only one language for the interface and the computation. It would at least simplify the interconnection between the interface and the data banks.

-It would surely make the code more readable. The Program Object Oriented (POO) language (C++ and JAVA) is simpler to understand because it allows more graphical representation of the code and variables are naturally grouped by concern in the objects.

-having access to the internet forums when encountering troubles while programming is really an asset for the new programmer. It is barely possible for Fortran now.

- Above all, the possibility to generate automatically documentation in the form of an html file is a huge asset, which would permit having a complete documentation for every new release of the program.

Doxygen for C++, or Javadoc for Java perform this kind of work. Universal Report and Doxygen are two programs, which could also generate documentation for Fortran.
The condition of these programs is that the programmers have to respect certain syntax to have it generating a meaningful documentation so none of them would be applicable directly on the PrADO code as it is now. Every file should be reworked so that it follows the program rules. Many experiments have been led with PrADO and Doxygen and the result was an index of the source files in HTML format. By clicking on the name of a source file, it only shows its contents with wrong colour code. The reason why it did not match is because the coding conventions did not match any of the possible configurations of Doxygen. The following pictures give an idea of the possibilities that Doxygen could provide to PrADO, and examples of correct automatic documentation can be found on the software website.

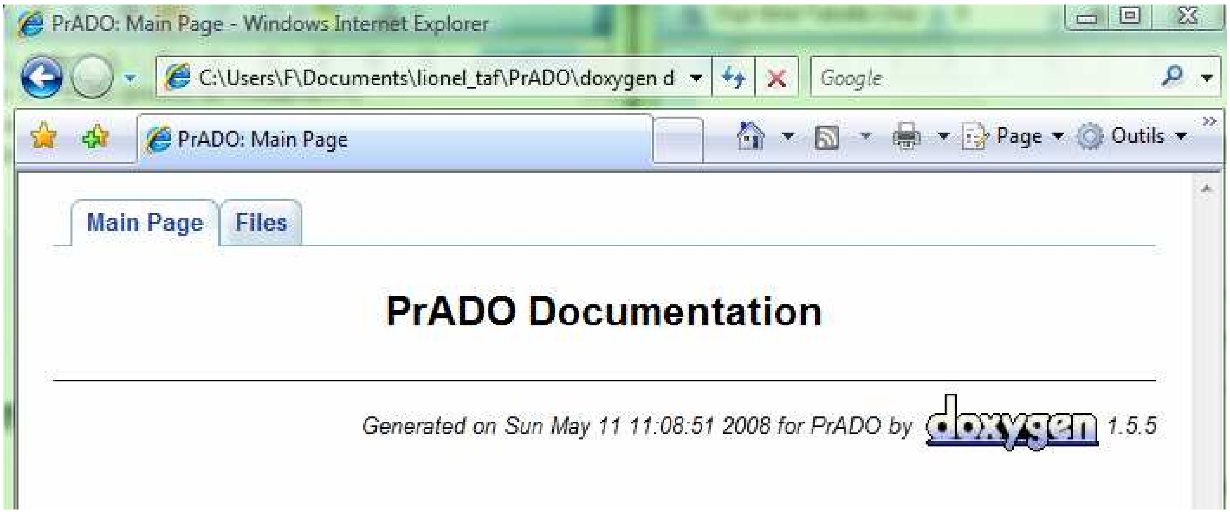

Figure 7.1 Main page of a html file generated with Doxygen

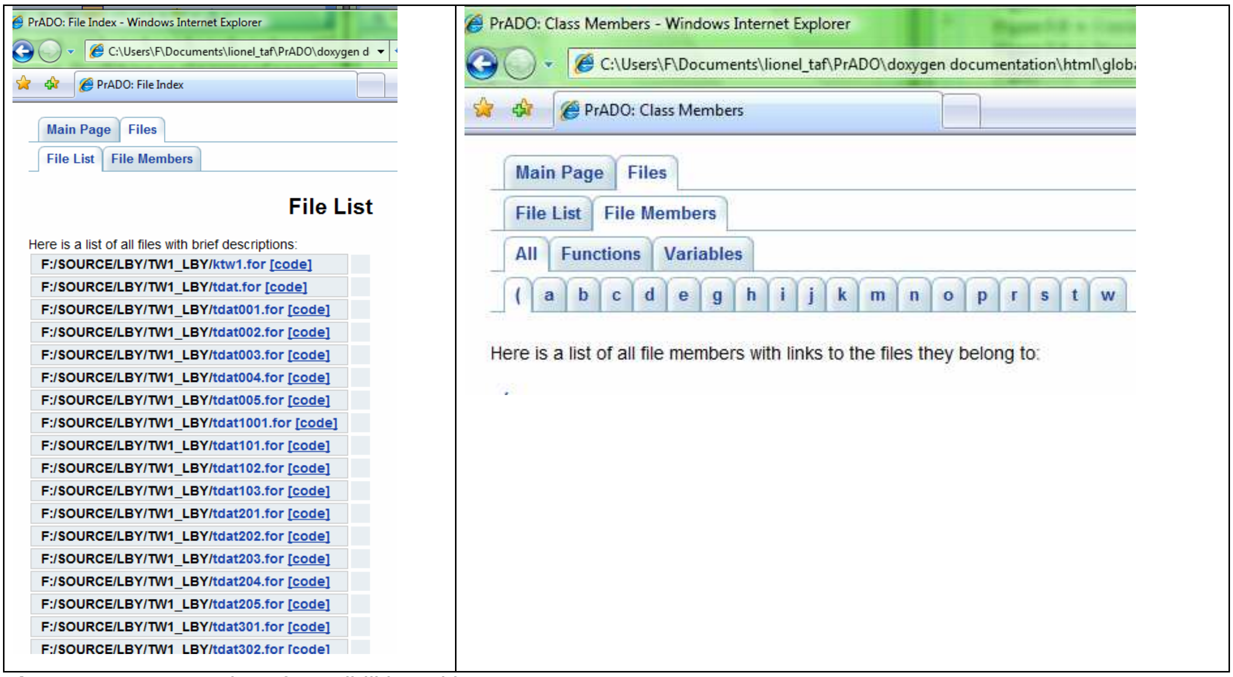

Figure 7.2 Examples of possibilities with Doxygen

Another possibility to make the documentation process automatic is to create a word model with a macro, which would produce a document out of the pre-existing syntax of PrADO. Not all files follow the exact same rules but some of them are more common: the description is usually between two lines of “-“, the chapters of the program are underlined by quotation mark (“), etc... Therefore, it would be possible to use this information to realize a macro filling a Microsoft Word model. This document could be then saved under an html format and the produced file could be called from an index with all the files names, as in Doxygen. The index could also link to other documents such as this report, **KIESEL 2007** or IFL’s own documentations, and the important files could be enhanced with formula translating the equation used or remarks.

This would be a possibility to make the documentation process half-automatic but the content of the documentation and the coding convention have to be decided by the programmers.

Moreover, the details in the PrADO file are written in German and an automatically generated documentation would be in German.

8. Conclusions

This project shows the principal capacities of PrADO and how to use them. The main window is especially detailed. It shows clearly PrADO as a very powerful aircraft design tool.

The structure of the program appears quite clear in terms of folders and files (p.23 to 30) and the scheme of interconnections between the various kind of files is very complex (p.30).
A study of the module's structure shows how the program is itself structured and it shows also partly how its main functions are accomplished. The study of the main libraries describes the tools which PrADO has at its disposal.

The study of the jet engine parts shows that if one would like to add a jet engine model in the structure of PrADO he will have to change many files to add really the possibility of a new jet engine kind or he can work on the pre existing files to make them give the performance of the new jet engine. What is not described here is how to create the geometry for a new jet engine. I hope this part will be useful for the next student wishing to add a turbofan, for instance.

To conclude this report, I hope that the methodology will be helpful for anyone desiring to understand what is behind PrADO and I hope that the discussion will be interesting for the people coding this powerful program.

Acknowledgements

First, I would like to thank my girlfriend and my parents for all their support during my studies and especially during this stay in Germany. Without your appreciation and assistance, I could never have done this semester in Germany.

Then, I would like to thank my supervisors in Germany and in France, Mr François Stephan, Farida Chabouni, Prof. Dr. Scholz, and Kolja Seeckt who always answered my questions very helpfully. Furthermore, I am fully aware that doing this report has been a great opportunity to me and was only possible because of your personal advice.
Thanks also to Mr Heinze who gave me hints to start my research.

And last but definitely not least, there are Dominik and Chloe, who very often reminded me of more reasons for being in Germany than just writing my report.

References

Heinze 2008:
HEINZE, Wolfgang: *PrADO-TA2*. Braunschweig: Institut für Flugzeugbau und Leichtbau, 2007

Scholz 1999
SCHOLZ, Dieter: *Skript zur Vorlesung Flugzeugentwurf*. HAW Hamburg, Fachbereich Fahrzeugtechnik und Flugzeugbau, Vorlesungsskript, 1999
URL: http://www.haw-hamburg.de/pers/Scholz/Flugzeugentwurf.html

TECPLOT
URL: http://www.tecplot.com (2008-05-16)

D**oxygen**
URL: http://www.doxygen.org (2008-05-16)

Universal Report
URL: http://www.omegasystems.org (2008-05-16)
Kiesel 2007
KIESEL, Torsten: *Methodisches Entwerfen von Verkehrsflugzeugen mit PrADO*. HAW Hamburg, Fachbereich Fahrzeugtechnik und Flugzeugbau,
URL: http://www.haw-hamburg.de/pers/Scholz/Flugzeugentwurf.html

Seeckt 2006
SEECKT, Kolja: *Performance assessment of part-electric* HAW Hamburg, Fachbereich Fahrzeugtechnik und Flugzeugbau
URL: http://www.haw-hamburg.de/pers/Scholz/Flugzeugentwurf.html