A Preliminary Aircraft Design Process

Developments at DLR

Martin Hepperle
Institute of Aerodynamics and Flow Technology
Braunschweig, Germany
DLR – Sites and Figures

DLR is the …
… German Aerospace Research Center
… National Space Agency of Germany

DLR in numbers
about 5600 employees
28 research institutes and facilities
13 sites
Offices in Brussels, Paris, Washington

DLR Budget (751 Mio € in 2010)
215 Mio € in aeronautics
≈ 50% Federal Republic of Germany
≈ 50% third-party-funds
Research Fields

Aeronautics

Energy

Transportation

Space
Conceptual & Preliminary Aircraft Design in DLR

We deal with a very wide range of scenarios, configurations, technologies and methods.
Preliminary and Multidisciplinary Design in DLR

- Multidisciplinary Design, Analysis and Optimization
  - The best aircraft is not a single discipline optimum but the best compromise.
  - MDO is a traditional component of preliminary aircraft design.
  - Individual technologies must be assessed at overall aircraft level.
- DLR Specifics
  - DLR consists of many institutes in different locations in Germany.
  - Regular collaboration with external partners.
  - In 2005 DLR started the first project for preliminary design capabilities
    - Project TIVA (Technology Integration for the Virtual Aircraft).
    - One outcome was the selection of a process automation tool.
  - Since then, several follow-on projects were building on this initial work.
It all began with TIVA...
(Technology Integration for the Virtual Aircraft)

• Objectives
  • To develop and to apply the capability to assess technologies in the context of the complete aircraft

• How
  • Create a network of multidisciplinary tools by linking DLR’s institutes, locations and experts.
  • Harmonization of data models and automation of data exchange.

• Results
  • Network of key experts for future projects.
  • Simplified and error free data exchange.
  • Direct the development of new technologies in all disciplines.
Data Exchange Format „CPACS“
Common Parametric Aircraft Configuration Scheme

• Describes the system „Aircraft“ or even “Air Transport System”
  • uses parametric models (if possible),
  • contains data for all disciplines and interactions.
  • details at preliminary design level (with extensions for HiFi),

• Application
  • all tools read and write CPACS data,
  • tools augment the data set with new results,
  • user can insert data as needed, e.g. experimental data.
Data Model – what to store and how?

- CPACS
- MySQL
- ASCII
- CATIA
- ModelCenter 9.0
- STEP
- XML

CPACS

Individual flight segments
- cruiser
- descent
- horizontal
- final approach

Parameters:
- fixed
- variable
Data Exchange Format „CPACS“
Common Parametric Aircraft Configuration Scheme

Requirement

- Hierarchical data storage
  - flexibility and extensibility
  - templates, catalogues, links

- Usage
  - automatic & manual editing
  - widely support

- Consistence and documentation
  - no redundant data
  - we use SI-units
  - documentation and validation

Extensible Markup Language (XML)

- tree-like structure
- central catalogues
- cross references via “UID”-attributes

- Extensible Stylesheet Language (XSL), XPATH and our support libraries
- XML-editors and ASCII editors
- human readable ASCII format

- Definition using „XML-Schema“
- Automatic Schema documentation
- Automatic data validation using Schema
Data Exchange Format „CPACS“
Common Parametric Aircraft Configuration Scheme

- Elements of the Air Transport System
- Catalogues of reusable elements
- Aircraft
  - geometric data
  - structural data
  - propulsion
  - aerodynamics
  - flight mechanics
  - more…

```
cpacs
  header
  vehicles
  missions
  airports
  fleets
  toolspecific

    cpacs
    aircraft
    engines
    profiles
    structuralElements
    materials
    composites
```

name
description
reference
fuselages
wings
engines
landingGear
systems
global
analyses
Data Exchange Format „CPACS“ – Example: Wing
Common Parametric Aircraft Configuration Scheme

- Each wing is composed of trapezoidal segments
  1. `<sections>`
     - references section from catalogue
     - relative placement of sub-elements
  2. `<positionings>`
     - position of each `<section>` relative to parent section `<section>`
  3. `<segments>`
     - defines wing segment spanned by 2 `<sections>`

- Wings are linked to a parent component
Data Exchange Format „CPACS“ – Example: Wing
Common Parametric Aircraft Configuration Scheme

- Automatic generation of multiple models:
  - Lists (XSLT)
  - 2D-Views
  - CAD Model Rhinoceros
  - CAD Model CATIA V5
  - VSAERO grids
  - BLWF grids
Data Exchange Format „CPACS“ – Documentation

Common Parametric Aircraft Configuration Scheme

- **CPACS-Definition as an XML-Schema**
  - Definition of tree structure and node contents.
  - Automatic validation of data sets using the CPACS-Schema
  - Documentation is a part of the CPACS-Schema
  - Automatic generation of online or printed documentation from CPACS-Schema

- **CPACS-Documentation**
  - Compiled HTML help file (Windows)
  - Improved documentation in progress
Support Tools – **TIVA XML Interface Library (TIXI)**

- Library to ease application of **CPACS** for software developers (engineers)
  - based on the free **Libxml2**
  - available for **Microsoft Windows** and **Linux**
  - Usable in most relevant programming languages, **FORTRAN, C/C++, JAVA, Python, ...**

- **Capabilities**
  - read and write **CPACS** data sets with inclusion of external data(!)
  - validation of **CPACS** data sets (**CPACS-Schema**, UID-references)
  - read and write nodes using **XPATH** expressions
  - high level functions to read and write of complete sub-trees (DOM)
Support Tools – **TIVA Geometry Library (TIGL)**

- **Library to use CPACS geometry data in software**
  - Based on **TIXI** and the **OpenCascade** CAD kernel
  - available for **Microsoft Windows** and **Linux**
  - Usable in most relevant programming languages, **FORTRAN**, **C/C++**, **JAVA**, **Python**, ...

- **Capabilities**
  - Builds a complete 3D model of the aircraft geometry (currently: wings and fuselages)
  - Functions to query geometry information (points, intersections, areas, volume, …)
  - Export of a closed volume model (**STEP** / **IGES** / **VTK** / **STL**)
  - Available for systems and software like **TIXI**
Support Tools – **TIGL-Viewer**

- **TIGL-Viewer**
  - Interactive visualization of the 3D CAD model of **TIGL**
  - Inspection of components (e.g. wing sections)
  - Additional functions (e.g. **STEP** / **IGES**-Export)
Data Exchange Format „CPACS“ – Wrapping
Common Parametric Aircraft Configuration Scheme

Tool wrapper
- interfaces tool input and output with CPACS
- extracts only the relevant parts from CPACS (a subtree)
- good solution for existing tools
  - no modification of the core tool
  - only a subset of CPACS must be handled

Direct CPACS interface
- Tool reads and writes CPACS
- good solution for new tools
- additional results can be stored in additional files or as part of tool-specific CPACS extensions
### Tools with CPACS interface and MC wrappers

<table>
<thead>
<tr>
<th>Category</th>
<th>Tool</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handbook methods</td>
<td>VAMPzero</td>
<td>LY</td>
</tr>
<tr>
<td></td>
<td>HandbookAero</td>
<td>AS</td>
</tr>
<tr>
<td>Aerodynamics</td>
<td>LiftingLine 1 / POLINT</td>
<td>AS</td>
</tr>
<tr>
<td></td>
<td>VSAERO</td>
<td>AS</td>
</tr>
<tr>
<td>Structures</td>
<td>BoxBeam</td>
<td>FA</td>
</tr>
<tr>
<td></td>
<td>PARA_MAM / S_BOT</td>
<td>FA, BK, LY</td>
</tr>
<tr>
<td></td>
<td>AC-CRASH 2</td>
<td>BK</td>
</tr>
<tr>
<td>Aeroelastics</td>
<td>MONA</td>
<td>AE</td>
</tr>
<tr>
<td></td>
<td>ELWIS 3</td>
<td>LY</td>
</tr>
<tr>
<td>Propulsion</td>
<td>TWDat / GTlab 4</td>
<td>AT</td>
</tr>
<tr>
<td>Landing Gear</td>
<td>LGConcept</td>
<td>AE</td>
</tr>
<tr>
<td>Mission Simulation</td>
<td>TCM</td>
<td>LY</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>RM</td>
</tr>
<tr>
<td>T/O LDG</td>
<td>MAPET</td>
<td>FT</td>
</tr>
<tr>
<td>Flight Dynamics, Loads</td>
<td>FDS 5</td>
<td>RM</td>
</tr>
<tr>
<td>Handling Qualities</td>
<td>HAREM</td>
<td>FT</td>
</tr>
<tr>
<td>Cabin PreDesign</td>
<td>FuMe 6</td>
<td>LY</td>
</tr>
<tr>
<td>Boarding Simulation</td>
<td>TOMICS</td>
<td>FW</td>
</tr>
<tr>
<td>Costs</td>
<td>LCC</td>
<td>LY</td>
</tr>
<tr>
<td></td>
<td>DOC</td>
<td>LY</td>
</tr>
<tr>
<td>Over-all Noise</td>
<td>PANAM</td>
<td>AS</td>
</tr>
<tr>
<td>Engine Noise</td>
<td>HEIDI</td>
<td>AT</td>
</tr>
<tr>
<td>Climate Impact</td>
<td>AirClim</td>
<td>PA</td>
</tr>
<tr>
<td>ATM simulation / mission</td>
<td>TrafficSim</td>
<td>FL</td>
</tr>
<tr>
<td>Radiation</td>
<td>RADIATION</td>
<td>ME</td>
</tr>
<tr>
<td>IR Signature</td>
<td>SIGMA</td>
<td>MF</td>
</tr>
<tr>
<td>Radar Signature</td>
<td>NIRATAM</td>
<td>HR</td>
</tr>
</tbody>
</table>
Typical Workflow with CPACS components

Start → ModelCenter
- objective converged?
  no → Engine Position
  no → Wing Position

End

CFD (TAU) → Mesh (Centaur) → CAD (Catia)

- Tail Plane Sizing
- Weight, Fuel,
- Center of Gravity

Δε_{HTP} → static margin

Δα_{a/c}
Typical Workflow with CPACS components

1. **Import initial CPACS data set**
2. **Insert design variables**
3. **Aerodynamic performance and loads**
4. **FE-model creation, sizing and analysis**
5. **Extract objective value(s)**
National CPACS Symposium
(March 2012 at DLR Hamburg)

- Goal
  - Start a national dialogue with the stakeholders of preliminary aircraft design capabilities (DLR, universities, industry)
- Questions and Answers
  - Which design projects are of interest?
  - Which tools are developed and applied?
  - Which capabilities are where?

- Workshop for DLR Open Source components:
  - CPACS data model
  - TIXI, TIGL support libraries
  - CHAMELEON@RCE alternative framework
  - VAMPzero conceptual design

Participants:
- Airbus
- Cassidian
- EADS-IW
- Bauhaus Luftfahrt
- RWTH Aachen
- TU Berlin
- TU Braunschweig
- HAW Hamburg
- TU Hamburg-Harburg
- Uni Linköping
- TU München
- Uni Stuttgart
- DLR LY, SC, AS, AE, RM, FT
ModelCenter and AnalysisServer Extensions

- CPACS
  - import/export/merge components

- Other (freely available from DLR, resp. PHX Solution Archive)
  - Optimization algorithms in AnalysisServer
  - CSV/Excel Export (directly via COM into Spreadsheet)
  - OctaveWrapper (Matlab compatible, Open Source)
  - Tecplot Export (e.g. for 2D parameter studies)
  - FileWrapperXML (FileWrapper plus XML read/write capabilities)
Integration of Optimization Algorithms into AS

- Integration of remote optimization tools
  - CONMIN, BFGS and ADS are legacy FORTRAN codes.
  - Interface to AnalysisServer is defined in Java.
  - Java’s javah tool creates a C++ interface (Java ↔ C++).
  - FORTRAN can be called from C++ (C++ ↔ FORTRAN).

- Drawbacks
  - Prescribed interface and calling logic requires optimizer implementation with “reverse communication”.
  - A crashing optimizer can kill AnalysisServer.
Extensions – Components for Analysis Server

- Optimization Components
  - Unconstrained Optimization (in Java)
    - Regula Falsi, 1D-Newton Iterator
    - Classical Simplex
    - Particle Swarm
    - Simulated Annealing
    - Broyden-Fletcher-Goldfarb-Shanno

- Constrained Optimization (via JNI)
  - ADS (NASA, Vanderplaats, FORTRAN)
  - CONMIN (NASA, Vanderplaats, C)
Plug-Ins for DataExplorer – CSV/Excel Exporter

- CSV export in DataExplorer is limited (decimal character, separator)
- New Excel Export Plug-In
  - writes data to file, or
  - can send data directly to Excel (via COM),
  - some customization options.
Plug-Ins for DataExplorer – Tecplot Exporter

- Graph options in DataExplorer are limited (sufficient for its purpose)
- Sometimes a better graph quality is required, Tecplot is widely used.
  - Plug-In saves data to file in Tecplot Format
  - well suited for 2D data
  - 3D data need some manual mods. concerning IJK-dimensions
Extended FileWrapper – FilewrapperXML

---

### File Wrapper ExML example

#### @author: Martin Hepperle

#### @description: Shows how to use the FileWrapperExML.

#### @version: 1.3, 8 JUNE 2012

---

downloadInputs: true

RunCommands

```plaintext
RowFieldInputFile inputFile

RowFieldOutputFile outputFile

XMLInputFile xmlInFile

XMLOutputFile xmlOutFile
```

---

Example

---

regular FileWrapper I/O definition blocks

additional blocks

XMLInputFile

XMLOutputFile
FilewrapperXML

RunCommands
{
    # NOT! generate xmlInFile, is done automagically
generate inputFile

    # execute analysis
run "do_something.exe"

    # NOT! parse xmlOutFile, is done automagically
parse outputFile
}
```xml
<xml version="1.0" encoding="UTF-8" />
<vlm>
  <aero>
    <alpha>10.0</alpha>
  </aero>
  <gamma>10.0</gamma>
  <gamma>20.0</gamma>
  <gamma>30.0</gamma>
  <gamma>40.0</gamma>
  <gamma>50.0</gamma>
  <names>
    <name>name #1</name>
    <name>name #2</name>
  </names>
  <characters>
    <character>65</character>
    <character>66</character>
  </characters>
</vlm>
```

```
XMLInputFile xmlInFile {
    templateFile:      template.vlm_in.xml
    fileToGenerate:     "vlm in.xml"

    variable: ALPHA double     xpath="/vlm/aero/alpha"
    >>>   description="angle of attack"
    >>>   lowerBound=-1.0 upperBound=12.1 units="deg"

    variable: GAMMA double[] xpath="/vlm/gamma"
    >>>   description="angle array" upperBound=52.2

    variable: NAME  string[] xpath="/vlm/names/name"
    >>>   description="name array"

    variable: CHARS string[] xpath="/vlm/characters/character"
    >>>   description="enumerated string array"
    >>>   enumAlias="A,B,C" enumValues="65,66,67"
}
```
FilewrapperXML

XMLOutputFile xmlOutFile
{
  fileToParse: vlm.xml

  # Get the contents of ALL nodes below /vlm
  variable: All    string[] xpath="/vlm/"

  variable: Cfx    double   xpath="/vlm/aero/cfx"

  # Some XPath functions
  variable: FC     integer   xpath="count(/vlm/flap)"
  variable: D_Sum  double    xpath="sum(/vlm/flap/delta)"
  variable: D_3    double    xpath="/vlm/flap[last()]/delta"
  variable: D_1_2  double[]  xpath="/vlm/flap[position()<last()]/delta"
  >>> xpath="/vlm/flap[position()<=last()]/delta"

  variable: DLast  double    xpath="/vlm/flap[\${FC}]/delta"
}
GeomScript – Implementation Example: Tecplot

ModelCenter

.../GeomScripts

tecplot.bsh

.tecplot.jar contains class “tec” which implements

static public String getExt()
static public String getName()
static public double[] getBB()
static public void render(GL gl, String path)

• BeanShell implementation well suited for prototyping.
• Compiled Java implementation better suited for performance.
• Additional classes used for debugging.
GeomScript – Minimum BeanShell interface

```
/** Display of FLOWer Tecplot files */
addClassPath(dir + "\\GeomScripts\\tecplot.jar");
import tecplot.tec;

String getExt()
{ return tec.getExt(); }

String getName()
{ return tec.getName(); }

void render(String path, GL gl)
{
   tec.render(gl, path);
   double[] box = tec.getBB();
   bbMinX = box[0]; bbMaxX = box[3];
   bbMinY = box[1]; bbMaxY = box[4];
   bbMinZ = box[2]; bbMaxZ = box[5];
}
```

`tecplot.bsh` implements interface to ModelCenter

`tecplot.jar` contains class “tec” with implementation
GeomScript – Test Harness for Development

- Debugging with BeanShell and ModelCenter not practical.
- Test harness for development and debugging was required.
- Consists of a Java main program which calls
  \[ \text{static public void render(GL gl, String path)} \]
- Mouse Interface for arcball rotation, scaling, translating.
- Tested and debugged code can be transferred directly to GeomScript.
- Difficulties
  - Some trial and error to set up the OpenGL context like ModelCenter.
  - How to implement a pure 2D-mode for x-y graphs?
Summary

- DLR has developed CPACS
  - the core components are:
    - data model for preliminary aircraft design (CPACS),
    - set of support libraries (TIXI, TIGL).
  - CPACS is available since 2012 as an Open Source project.
  - CPACS can be used with ModelCenter and/or DLR‘s RCE.

- DLR has developed several components for ModelCenter/AnalysisServer
  - CPACS reader/writer/merger (ModelCenter),
  - CPACS wrapper (AnalysisServer),
  - Several smaller Plugins, wrappers and GeomScripts.
- Future Work will create bridges between ModelCenter and HiFi specific processes (running mostly in EADS FlowSimulator)