Automation of Aircraft Pre-Design with Chameleon

Arne Bachmann
Simulation- and Software Technology
German Aerospace Center (DLR)

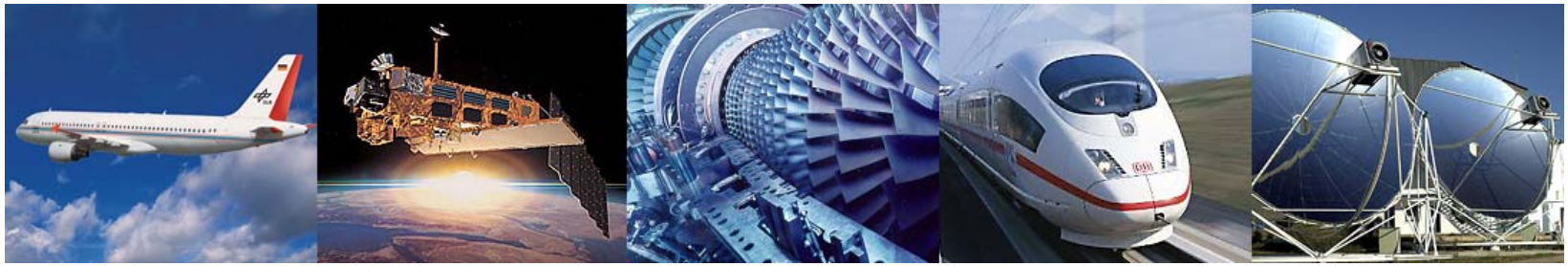
ADVCOMP 2009, Oct 13th, Sliema/Malta

Deutsches Zentrum
für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

Overview

- Who we are
- Introduction
- Exemplification
- Outlook

DLR
German Aerospace Center



- Research Institution
- Space Agency
- Project Management Agency

Locations and employees

6200 employees across
29 research institutes and
facilities at

- 13 sites.

Offices in Brussels,
Paris and Washington.

National and International Networking

Customers and partners: Governments and ministries, agencies and organisations, industry and commerce, science and research

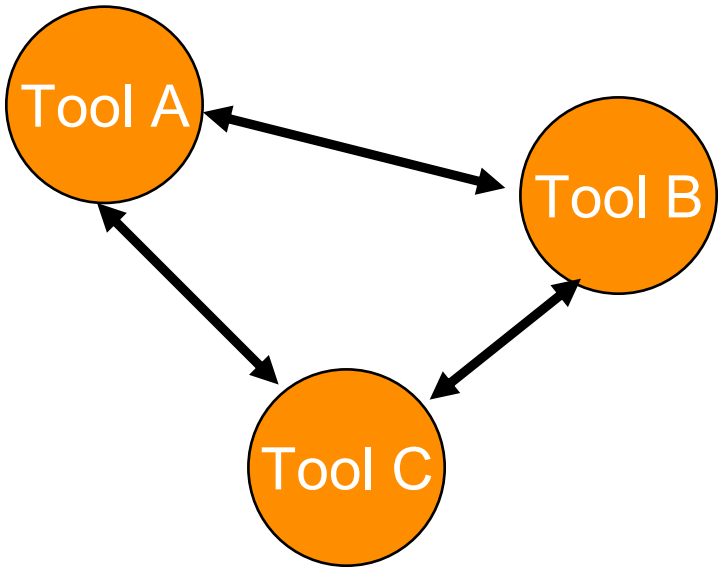
Motivation

- Airplane pre-design:
  - Simulate and evaluate new plane configurations
  - Test new flight procedures
  - Assess probable costs
  - Optimize for certain goals:
    - emission, capacity, efficiency
- Interdisciplinary:
  - Many disciplines, institutes, partners involved
    - Strong interdependencies
    - Close cooperation necessary
  - Looking for *global* optima

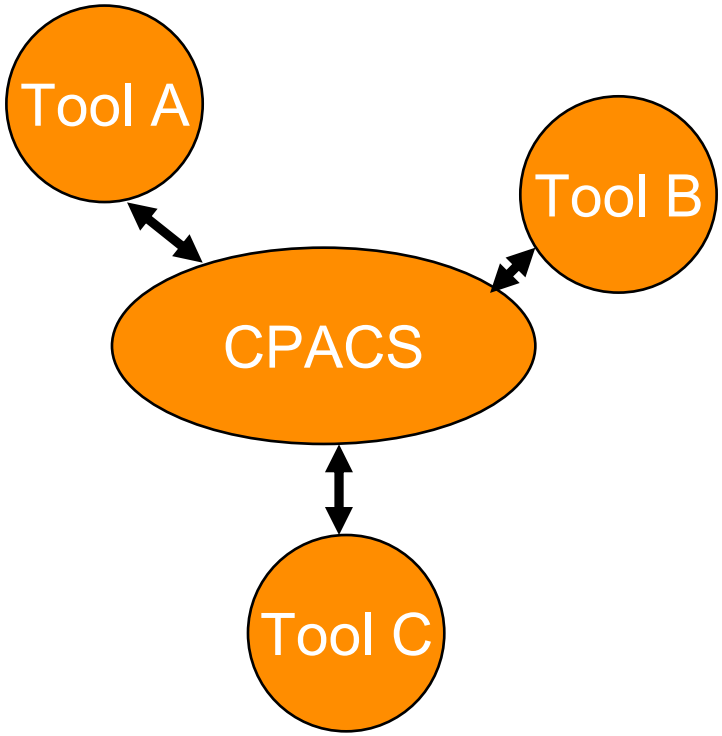
Collaboration

- Institutes already have *very* good optimizers for their domain problems!
- But:
  - They use their own or proprietary I/O formats
- Cooperation between institutes with their tools is taking place often!
- But:
  - Interfaces for data exchange are defined ad hoc
  - No common data format
  - No reusable automated process chains / workflows

Batch-processing drawbacks vs. Common format



N x (N-1) converters



2 x N converters

Use case

- Engineers collaborate in interdisciplinary projects
  - Share their expertise via problem-solving tools (e.g. simulation)
  - But don't give away their sovereignty in their research field
    - They simply provide a service with well-defined I/O (SOA)

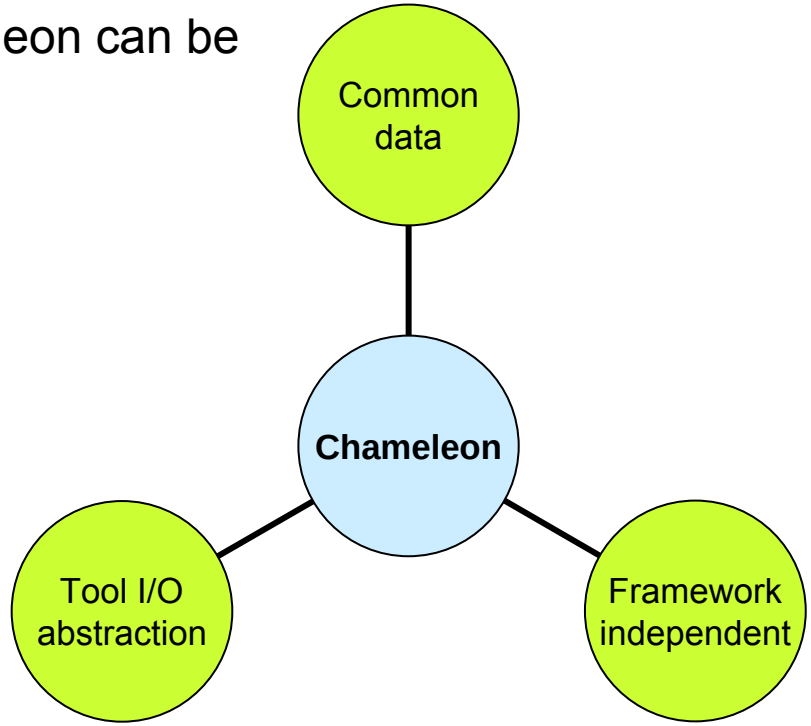
- For problem-solving, a researcher can combine the published tools
  - Simply by building a tool chain/workflow together from her computer's desktop
  - A framework takes care of all the infrastructural stuff
    - Service discovery
    - Configuration
    - Data flow, workflow, data interfacing, integration & visualiz.

Chameleon

- Why yet another framework?
  - Existing ones aren't flexible enough
    - With regards to flexibility of data connections between tools
    - With regards to infrastructure
    - With regards to user-guidance and simplicity
- Thus we put Chameleon *on top* of existing software integration systems
  - ModelCenter
    - http://phoenix-int.com
  - RCE
    - "Remote Component Environment"
    - http://rcenvironment.de

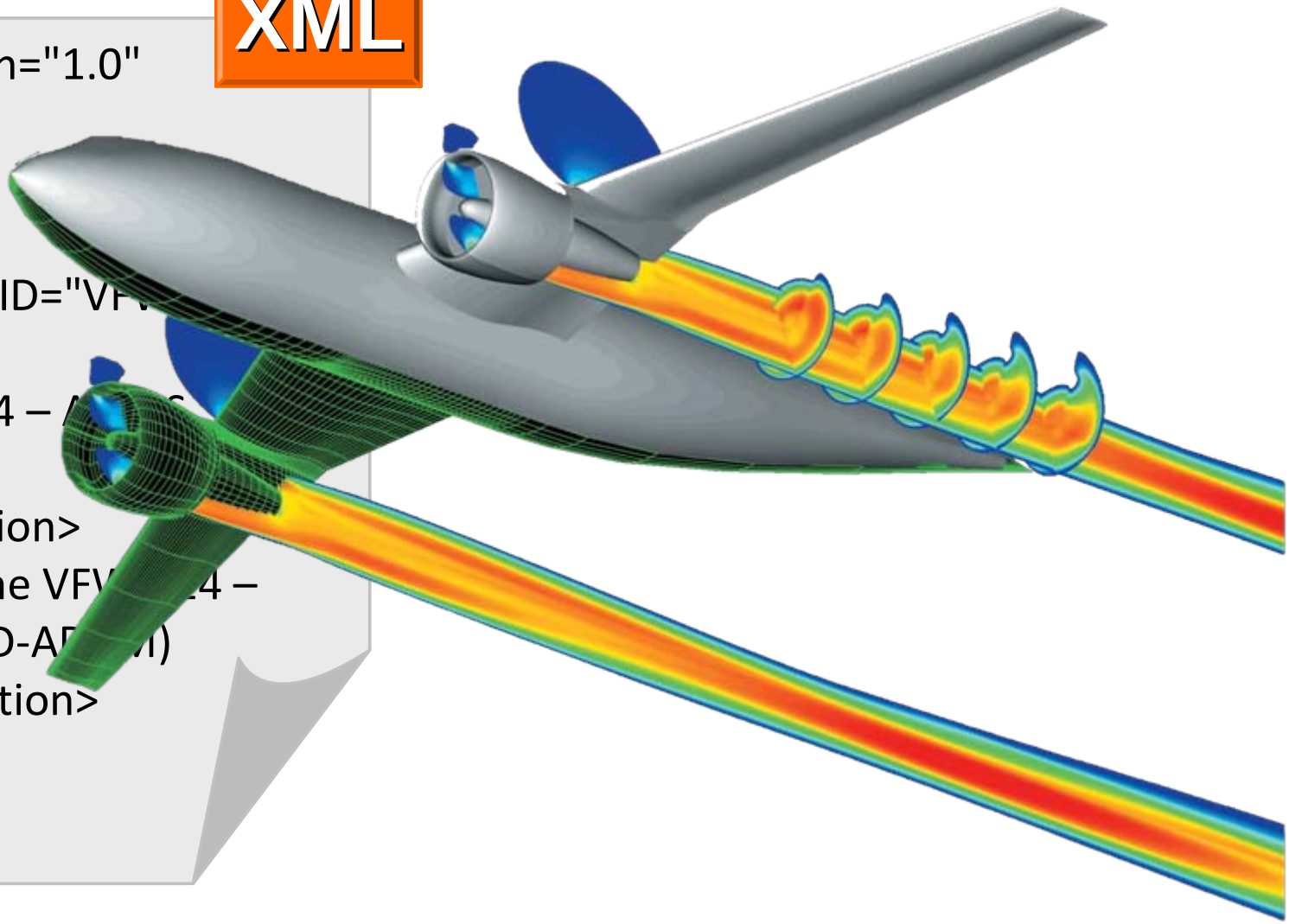
Chameleon

- Is a software suite with several abstraction layers
  - Data abstraction: Common data exchange format for all parties
  - Tool abstraction: Wrap proprietary tools and custom formats
  - Framework abstraction: Chameleon can be adapted to an(y) underlying software integration framework



XML

```
<?xml version="1.0"
<cpacs>
  <vehicles>
    <aircraft>
      <model uID="VF
      <name>
        VFW-614 – A
      </name>
      <description>
        This is the VF 4 –
        ATTAS (D-A )
      </description>
      …
</cpacs>
```



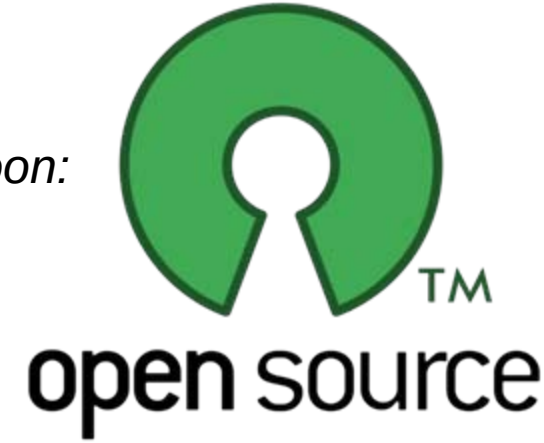
Data integration

Common Parametric Aircraft Configuration Scheme (CPACS)

XML-based data format

- Structured, extensible, transformable
- Hierarchical data structures

Soon:



Data concept

- Parametric description, several information detail levels storable
- Can be extended whenever new fields of science need to integrate

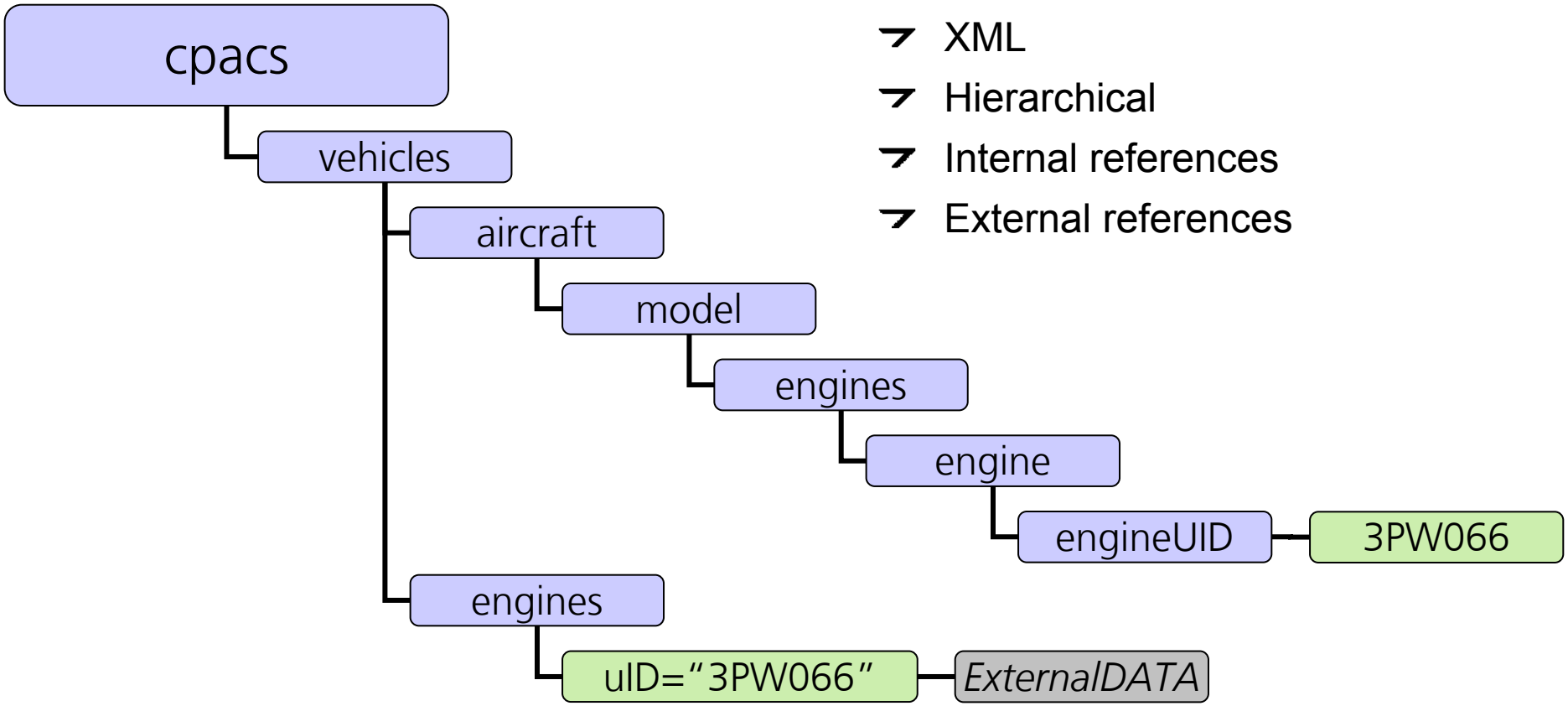
Dataset integrity by XML schema (XSD)

- XSD allows for automatic validation of datasets
- Integrated data format documentation within the schema → PDF/HTML

Data integration

Common Parametric Aircraft Configuration Scheme (CPACS)

- Basis for all applications
- XML
- Hierarchical
- Internal references
- External references



Tool wrapping component

- I/O converters from CPACS to custom XML I/O
  - Used by tools that have their own XML format
- Wrappers from proprietary formats to XML
  - Used when tools are unmodifiable (no source)
    - Because one doesn't own rights
    - Because they aren't supported any longer
    - Because it's easier to write a little wrapper
- This two-stage wrapping shields both tools and the common dataset definition from changes in the other
  - By providing a mapping mechanism for simple to complex cases

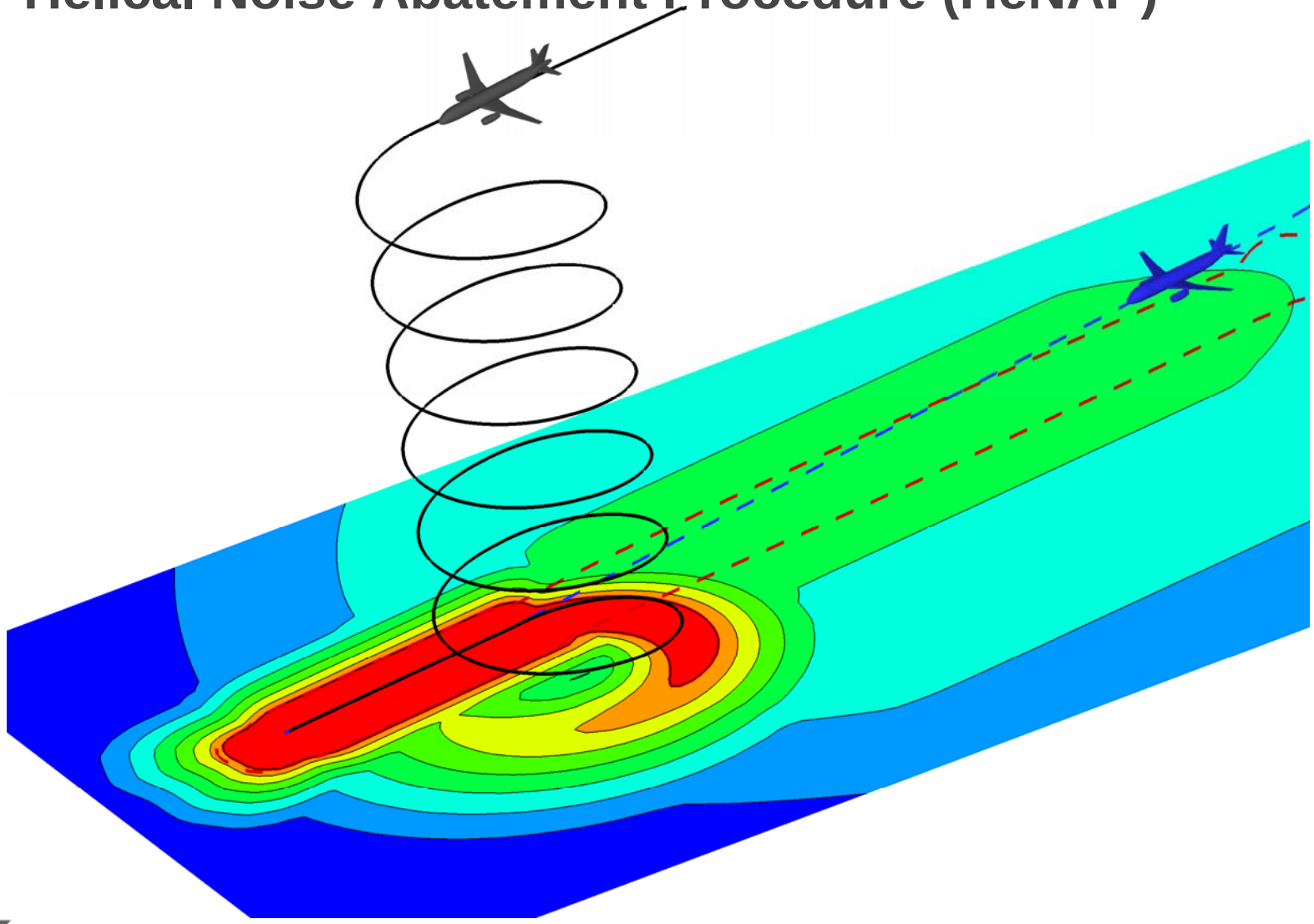
Framework abstraction layer

- Chameleon comes with useful libraries for
  - Simple XML access for wrapping tools, written in C (TIXI.lib/.dll)
  - Geometric library, written in C++ (TIGL.dll)
  - Interfaces for C, C++, Fortran & Python included
- Java GUI components for
  - simple import/export of CPACS data
  - visualization of airplane geometry from within the framework
- The combination of CPACS, ToolWrapper and Java components
  - make reusing the Chameleon suite in other frameworks easy
  - Under current development: JAR → OSGi; Swing → SWT

Example application with Chameleon

- Simulation of a new flight approach procedure
  - Approach the airport in a helix shape instead of a straight decline
  - Involves cooperation of institutes for *propulsion technology, aerodynamics and flow technology, robotics and mechatronics*
- Use the Chameleon framework on top of *ModelCenter* to combine necessary tools to a workflow
- Eventually, check the simulated results with a real flight experiment with the *Advanced Technologies Testing Aircraft System* (ATTAS)

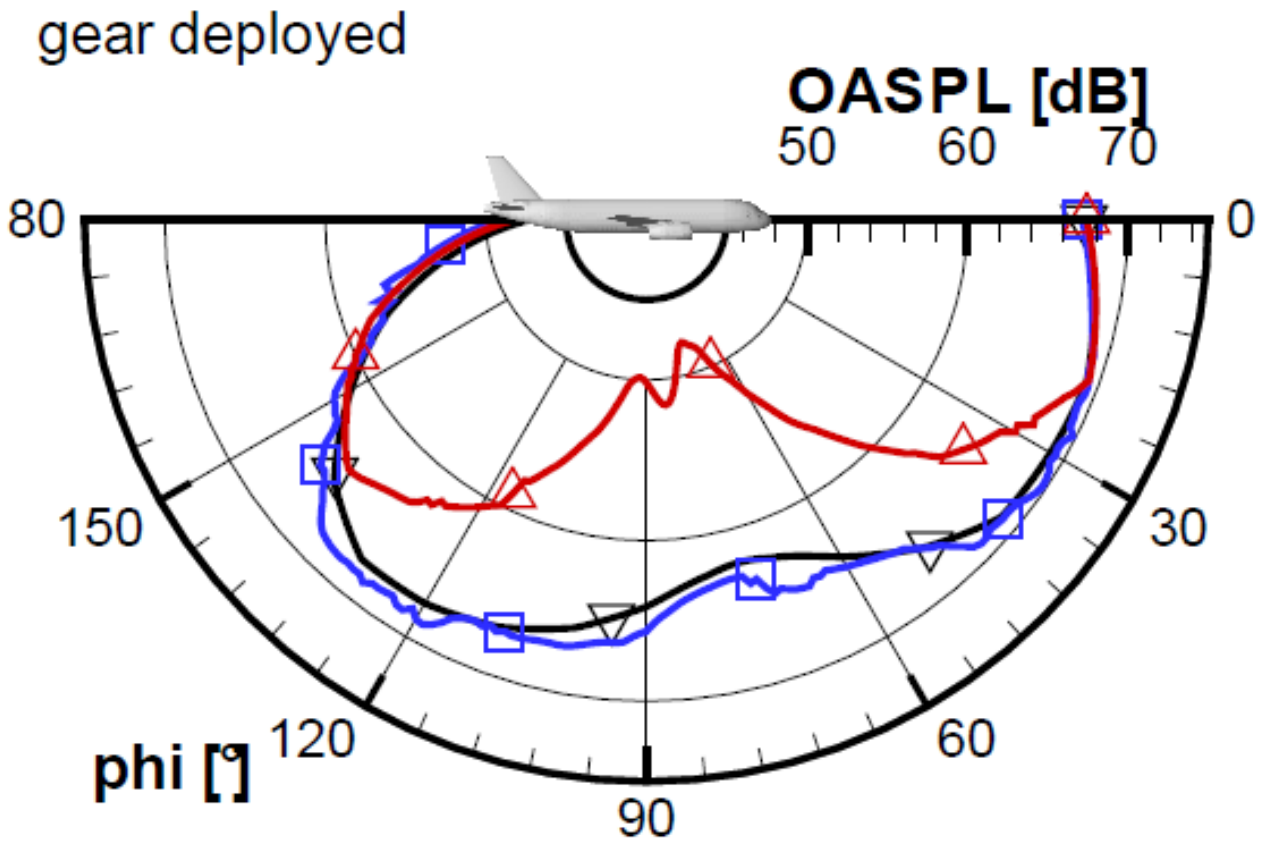
Helical Noise Abatement Procedure (HeNAP)



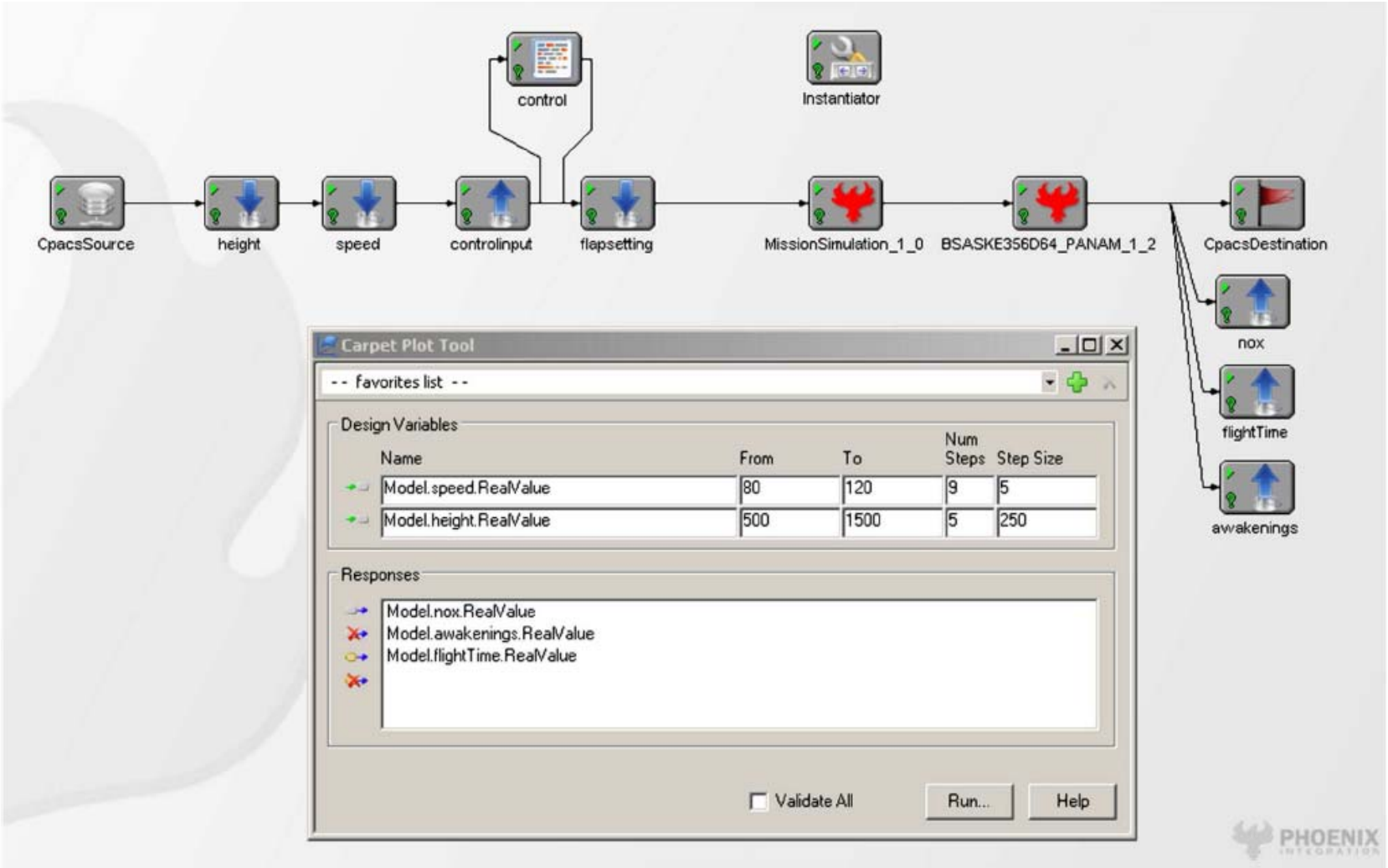
Example workflow: Tools involved

- Airplane geometry as input to the workflow
- Lifting Line:
  - Aerodynamics
- VarCycle:
  - Engine performance: thrust, fuel consumption
  - Emission data over mach + altitude (noise, NOx, COx)
- TWDat:
  - Database lookup for many existing engines
- PANAM:
  - Noise prediction tool
- SHADOW:
  - Noise shielding characteristics for airplane geometries

Example: Fan noise directivity



Example workflow



Verification of the simulation

Verification of the simulation

Conclusion

- Current drawbacks
  - No resilience features other than of the underlying framework
  - Same is true for monitoring (approximated percentages shown)
  - Ease of build-up/collaboration over pure performance
  - Parallelization only in workflow and on node/cluster
- Largest advantages
  - No fixed data connections between tools
  - Bunch of libraries to help engineers integrate and profit from Chameleon and CPACS
    - Simple tool wrapping
  - Quick build-up and easy sharing of new project workflows

Outlook

- Planned future tasks:
  - Include provenance data recording into our framework
  - Work on handling of large data sets
  - Integrate Chameleon with data management for CPACS datasets
  - Port Chameleon to the *remote component environment* http://rcenvironment.org

Questions?

http://www.walle-derfilm.de/