Applicability of OSS to Space Thermal Engineering
Open Source Software for Engineering Purposes

17th Thermal and ECLS Software Workshop
21. – 22. October Noordwijk ESA/ESTEC

Matthias C. Haupt
Reinhard Schlitt (OHB)
Frank Bodendieck (OHB)

Institute for Aircraft Design and Lightweight Structures
Institut für Flugzeugbau und Leichtbau
Technische Universität Braunschweig

m.haupt@tu-bs.de

Agenda

- Background
- Open Source Software
- Tool Integration
- Application
- Conclusions
## Therm-OSS Project:
- Application of the open source software (OSS) development model
- Thermal design and analysis tool
- ... made of OSS existing modules

## Therm-OSS Approach:
- Survey of suitable OSS for the tool and the development
- Development of the general architecture
- Implementation, test and ...

### Approach of the OSS survey:
- **What** we have looked for:
  - Functions for modelling and simulation
    - geometry, properties, grid,
    - mission and environment,
    - radiation, conductances,
    - solver, postprocessing
  - Engineering environment
    - scripting language, configuration control, graphical user interface
  - Development infrastructure
    - CASE (design and implementation)
    - configuration and installation
- **How** we have done the search:
  - collecting information available at team members and interested people
  - Searching in the internet at potential sites, e.g. http://sal.kachinatech.com/, http://sourceforge.net
  - look over the link collections of (OSS) internet pages
**Formal criteria of software selection:**
- Initiation of the project (age) and Version development (continuous development)
- License type
- Dependency on other software (language, OSS, non-OSS, complexity) and operation systems

**Functional criteria:**
- Provided functionalities
- Architectural features
- Algorithmic core (correctness)
- Documentation (theory reference, user manual, tutorials, examples, ...)
- Software quality (source code, configuration, version control)

**Subjective criteria:**
- Expected future of the project (continuation, user community, )
- Recommendation for Therm-OSS (suitable, integration)

---

**OSS: Geometry, Properties and Grid**

**Considered Tools:** (support of primitives and structured grids)

- **Ayam**
  - www.ayam3d.org
  - CSG, NURBS, RenderMan, tcl

- **Blender**
  - www.blender3d.org
  - Meshes, NURBS, animation, rendering, radiosity, Python

- **Giram**
  - www.giram.org
  - Primitives, CSG, POV, AutoCAD

- **innovation3d**
  - innovation3d.sourceforge.net
  - Meshes, (NURBS), (RIB)

- **K-3D**
  - k3d.sourceforge.net
  - Primitives, Meshes, CSG, RIB, animation, div. scripting

- **OpenCascade**
  - www.opencascade.org
  - Primitives, CSG, ... CAD

- **VRS**
  - www.vrs3d.org
  - Primitives, Meshes, Animation, library, Tcl-GUI

- **Chalmers**
  - www.na.chalmers.se:80/~andersp/chalmesh
  - 3D overlapping grid generator

- **Gmsh**
  - www.geuz.org/gmsh
  - 3D FEM grid generator, build-in CAD, post-processing

- **NETGEN**
  - www.hpfem.jku.at/netgen
  - 3D tetrahedral mesh generator, CSG, Brep(STL), OCC, solver

- **QMG**
  - www.cs.cornell.edu/home/vavasis/qmg-home.html
  - Geometric modelling(Brep), 2/3D mesh generator, solver, Tcl/Tk
OSS: Geometry, Properties, Grid

- k-3d
- Primitives
- Meshes
- Editing
- Material
- RenderMan

IFL

Institut für Flugzeugbau und Leichtbau, TU Braunschweig
### k-3d

#### UML diagram of Top-level

#### Uses
- GTK
- OpenGL
- XML

---

### Gmsh

#### FEM environment
- CAD
- grid generator
- Postprocessing

#### Uses
- FLTK
- OpenGL
OSS: Geometry, Properties, Grid

Gmsh

Grid examples
- Structured grid
- Unstructured grid

OSS: Mission and Environment

Considered orbit propagators:  

- Predict - Satellite tracking and orbit prediction program (SGP4,SDP4), additional tools (time converters, Calculate_Solar_position, Calculate_LanLonAlt, ...)
  www.qsl.net/ks2bd/predict.html

- GLSat - Satellite tracking and visualisation utility (SGP, SGP4)
  http://sourceforge.net/projects/glsat

- Project Pluto - Orbit predictor (SGP4/8,SDP4/8), additional tools (ALT_AZ, CLASSEL, EARTH2000)
  http://www.projectpluto.com/sat_code.htm

- ORSA - Orbit reconstruction, Simulation and Analysis framework for celestial mechanics.
  N-body integrators;
  http://orsa.sourceforge.net
OSS: Mission and Environment

△ Predict
△ Real-time satellite tracking and orbital prediction information
  ● System console
  ● Command line
  ● Network socket

△ Used by
  ● NASA: Goddard Spaceflight Center
  ● US Naval Research Laboratory
  ● Interferometrics: AMRAD-OSCAR-27
  ● SUNSAT-OSCAR-35 Satellite Command Team
  ● Stanford University’s Space System Development Laboratory (SSDL)
  ● Caltech: aligning radio telescopes against the sun position

Gsat client connected to Predict

Institut für Flugzeugbau und Leichtbau, TU Braunschweig

OSS: Environment and Loads

Considered radiosity tools: (DEIMOS Space contribution)

△ POV-Ray - Renderer for photo-realistic images, radiosity method similar to Radiance; http://www.povray.org
△ Radiance - Radiosity approach for lighting simulation, (view dependent); lloyd.lbl.gov/radiance.
△ Blender - Integrated modeling and rendering; (hard to extract the radiosity module); http://www.blender.org
△ Glutrad - simple and fast radiosity engine; http://www.cix.co.uk/~colceck/glutrad.htm
△ Raddoom - Java code of dynamic surface subdivision radiosity; http://perso.wanadoo.fr/psychomat/logiciels
△ Radiator - Matrix, progressive and wavelet radiosity with clustering; http://www-2.cs.cmu.edu/afs/cs/user/ajw/www/software/
△ RenderPark - Various ray-tracing and radiosity algorithms; (Support?) www.cs.kuleuven.ac.be/cswis/research/graphics/RENDERPARK

Institut für Flugzeugbau und Leichtbau, TU Braunschweig
Radiance

Lighting Simulation and Rendering System

Actual lighting for Z1 target array capture (video downlink)

Predicted lighting for Z1 target array capture

Considered FEM/FDM solver: (support of heat transfer / structured grids)
- Deal.II - library for adaptive FEM with error estimation http://www.dealii.org/
- Feat - System for FEM (domain description, mesh generation, shape functions, loads, boundary conditions, solution, error evaluation, graphical postprocessing http://www.featflow.de/
- MODULEF - library for FEM (automatic mesh generation, loads, boundary conditions, … ) http://www-rocq.inria.fr/modulef/english.html
- Mouse - OO framework for FVM on unstructured grids http://fire8.vug.uni-duisburg.de/MOUSE/
- Ofeli - library of finite element C++ classes http://ofeli.sourceforge.net/
- Tochnog - FEM system, http://tochnog.sourceforge.net/
OSS: Conductances and Solution

- Tochnog
- Commercial support by FEAT (NL)
- GPL.

- Equations:
  - Convection-diffusion
  - Stokes and Navier-Stokes
  - Elasticity.
  - Elasto-Plasticity.
  - Hypo-Plasticity.
  - Damage.
  - Thermal stresses.
  - Hypoelasticity.
  - Viscoelasticity.
  - Viscosity.
  - Wave equation.

3D thermal analysis of a sphere.

Residues on the adaptively refined grid.

Considered general solvers:

- LASPack - Iterative solver package (CGN, GMRES, BiCG, QMR, CGS, BiCGStab), multilevel, multigrid; [www.tu-dresden.de/mwism/skalicky/laspack/laspack.html](http://www.tu-dresden.de/mwism/skalicky/laspack/laspack.html)

- PETSc - Krylov subspace methods, Newton-based nonlinear solvers, time stepping (ODE) solvers; [www-unix.mcs.anl.gov/petsc/petsc-2](http://www-unix.mcs.anl.gov/petsc/petsc-2)

- SUNDIALS - Solver for initial value problems for ordinary differential equation systems and nonlinear algebraic systems. [www.llnl.gov/CASC/sundials](http://www.llnl.gov/CASC/sundials)
Considered plotting tools:

- **Chaco** - presentation quality scientific 2D graphics on a variety of output devices, http://www.scipy.org/site_content/chaco
- **Gnuplot** - command-driven interactive function plotting program, http://www.gnuplot.info/
- **Grace** - WYSIWYG 2D plotting tool for the X Window System and M*tif. (export of eps, pdf, MIF, SVG, PNM, jpeg, png; FFT analysis; curve fitting) http://plasma-gate.weizmann.ac.il/Grace/
- **PLplot** - library of scientific plot functions, can be used from C, C++, FORTRAN and Java, and Octave, Python, Perl and Tcl; http://plplot.sourceforge.net/.
- **VisAD** - Java component library for interactive and collaborative visualization and analysis of numerical data. https://sourceforge.net/projects/visad/.

Considered 3D visualisation tools:

- **OpenDX** - software package for the visualization of scientific, engineering and analytical data. Motif and X GUI, enhanced data model, advanced execution environment; http://www.opendx.org/
- **VTK** - C++ class library for 3D computer graphics, support for Tcl/Tk, Java, and Python; http://www.vtk.org

---

OSS: Plotting tools

- **Chaco** is a platform independent plotting library for Python
- Enthought, Inc. for the Space Telescope Science Institute (STScI)
- BSD license.
- Graphics toolkits:
  - wxPython
  - Tkinter
  - OpenGL
  - PDF
  - SVG
  - PIL
OSS: Plotting tools

- **Chaco** is a platform independent plotting library for Python
- Enthought, Inc. for the Space Telescope Science Institute (STScI)
- BSD license.
- Graphics toolkits:
  - wxPython
  - Tkinter
  - OpenGL
  - PDF
  - SVG
  - PIL

OSS: Tool Integration

Three considered Integration Environments:

- **SCIRun** - Problem Solving Environment; http://software.sci.utah.edu/scirun.html
- **XCAT** - Common Component Architecture (CCA) of the Indiana University, Extreme Lab; http://www.extreme.indiana.edu/xcat/
- **ifls** - A integration environment build of OSS components, based on Python and VTK
OSS: Tool Integration

- **SCIRun** is a Problem Solving Environment
- Computational Workbench
  - Rapid Prototyping
  - Module Development Tools
  - Extensible: Data, Algorithms
- Visual Programming
  - Computational Steering
  - Dataflow Interface
- Modeling, Simulation and Visualization
  - Model Construction and Manipulation
  - Numerical Approximation and Solution of PDE's
  - Scalar, Vector, and Tensor Field Visualization
- High-Level Utilities
  - Scene Graph and Widget Libraries
  - Math, Geometry, and Field Libraries
- Low-Level Utilities
  - Thread, Memory and Task Management

Institut für Flugzeugbau und Leichtbau, TU Braunschweig

---

OSS: Tool Integration

- **SCIRun** is a Problem Solving Environment
- Example: Bioelectric Field Simulation

Institut für Flugzeugbau und Leichtbau, TU Braunschweig
OSS: Tool Integration

**Ifls** – Objectives of our development:

- Automation and integration of the workflow
- Flexible data transfer between tools
- Integration of different source codes as well as (commercial) executables
- Simplicity, clearness, flexibility and robustness of the integration approach
- Efficient rapid prototyping for new (sub-)tasks
- Interoperability with project partners
- No monolithic tool but building blocks (build your own application)

We develop methods to solve our engineering problems

- Use of available technologies, standards and tools

  . . . suitable **Open Source Software** (OSS)

---

Application: Airplane Design

**Task in Airplane Design and Analysis:**

- Generic design with parametric geometry
- Use of proper design tools (e.g. CAD)
- Direct link to simulation and analysis codes with feedback
  - Derivation of corresponding analysis models (e.g. FVM, FEM, MBS)
  - Flexible postprocessing … system properties
- Mono- / Multidisciplinary studies in one environment
- Optimization w.r.t. various criteria and arbitrary design variables
- Integration into a Product Data Management Environment
- (Incorporation of knowledge based methods)
OSS: Tool Integration

Basic components of **ifls**:

- **Python**
  - Object-oriented scripting language
  - Contains elements of traditional languages
  - Nice, simple syntax
  - Modular structure
  - Great number of books
  - Unix, Windows, ... very stable
  - Scientific computing
  - Increasing acceptance

Class `MyClass`:

```
A simple example class
```

```
i = 123
def f(x):
    if x > 0:
        return 1
    else:
        return 0
```

- **Standard packages of Python**
  - *Tkinter*:
    - Widgets from Tk for GUI's
  - *Numerical*:
    - Vector / matrix objects
  - *Scientific Python*:
    - Scientific tools, MPI, NetCDF, Optimization, ...

- **Interface generators**
  - *pyfort*: Fortran
  - *swig*: C, C++

---

OSS: Tool Integration

- **Visualization Tool Kit vtk**
  - 3D computer graphic system, defines the architecture
  - compiled kernel C++
  - Wrapper for Tcl, Java and Python
  - Render windows, renderers, actors, properties, lights, cameras
  - Data objects (general, grid data), Process objects (source, filter, mapper)
  - Reference books and examples
  - OpenGL ... WinTel, Unix
  - I/O for VRML, IGES, 3DS, HDF, ...
OSS: Tool Integration

Visualization pipeline of **vtk**

- **Data Objects**
  - Represent information
  - Methods to create, access and delete this information
  - Methods to obtain characteristic features

- **Process Objects**
  - Operates on input data to generate output data
  - Sources interface to external data (Reader) or generate data from local parameters
  - Filter require one or more input data objects and generate one or more output data objects
  - Mapper objects are used to convert data into graphical primitives

**Pipeline Execution**
- Causes process objects to operate
  - Implicit control implemented demand-driven
  - Process object execution if input change
  - Two-pass process: update and execution

Integration environment **ifls**

- **Extension modules**
  - **utk** connects **vtk** and pure **Python** objects to maintain pipeline mechanisms
  - **usr** extends the capabilities of the **vtk**-process objects
  - **dtm** with special data and process objects for geometry and grid generation based on **DTURBS (IGES)** and **GridLib**
  - **stk** with process and import/export objects for analysis and simulation in a distributed environment
    (Structure: **Ansys**, **MSC**, **Abaqus**, Fluid: **HISSS**, **Flower**, **Cavecats**, **Tau**)

- **Graphical editor**
  - Interactive manipulation and visual programming
  - Analyzes the programmed object interactions/networks
  - Visualizes the object interactions (tree / graph)
  - Coding conventions for the automatic generation of the networks, object editors and documentation (html, latex, postscript, pdf).
  - **Python** codes are executable without the graphical editor in batch mode, because the GUI is an optional feature.
Application: Oblique Flying Wing

- Geometry by DTNURBS
  - wing, fin, wake
  - *.db (PrADO)
- Grid by GridLib
  - *.msh (HISSS)
- Results
  - *.sca (HISSS)
  - *.plt (Tecplot)
- Aerodynamic Loads
- Derivatives
- Optimization
- Interactive variations
- Student projects
**Application: Aeroelasticity**

- Reference geometry
- Euler code
- MSC/Nastran
- Coupling iteration for the equilibrium state

Deflection transfer by `vtkProbeFilter` and self-developed hybrid techniques

Institut für Flugzeugbau und Leichtbau, TU Braunschweig

**Application: Airport Environment**

- Flight dynamic simulation with `JSBSim` from the `FlightGear`-Project

- Complete set of the equation of motion including ground forces and FCS

- **XML**-formatted description of
  - Geometry
  - Massprops
  - Aerodynamics
  - LandingGear
  - Propulsion
  - Initial state

Servoelasticity

Institut für Flugzeugbau und Leichtbau, TU Braunschweig
Flight dynamic simulation with JSBSim from the FlightGear-Project

Complete set of the equation of motion for the simulation in the time domain

Class diagramm (end of 2001)

C++ implementation
Wrapped by swig
XML generator for HISSS aerodynamics
maneuver loads
script or interactive use

Therm-OSS - Statement of work:

“The system to be developed, will be able to perform the complete thermal analysis of a spacecraft, or part thereof. This includes:

- the definition or modification of a model of the spacecraft or component
- the definition or modification of a model of the environment;
- the definition or modification of a model of the mission and scenario;
- the definition and execution of the analysis, defined by the above steps;
- the evaluation or assessment of the results.

To build a model of a spacecraft, or component, the user will have available a catalogue building blocks (or objects). In this first version, the geometrical forms will be limited to a few simple ones, such as plane quadrilaterals, boxes, complete cylinders and complete disks. The building blocks and the assembly tree will be stored, together with all related thermo-physical properties in an object-relational database (e.g. PostgreSQL)”
Application: Therm-OSS

Proposed design:

Integration Framework with VTK Object Behaviour
Python interpreter

Institut für Flugzeugbau und Leichtbau, TU Braunschweig

Engineering infrastructure:

Extended u tk of ifls

ZopeDB, PostgreSQL

PyExpress

Institut für Flugzeugbau und Leichtbau, TU Braunschweig
Potential functional modules:

Integration of modules/tools as ProcessObjects:

- **VTKising:**
  - C++ programming: deriving classes from VTK abstract base classes, e.g. `vtkProcessObject`,
  - Python programming and the output will be a standards VTK data object using the `vtkProgrammableSource` class,
  - using the ifls extension utk, which is a reimplementation of some basic VTK classes in Python.
  
  The essential work is the implementation of the, `Execute` method, which do the input to output transformation.

- **Data Transforming:**
  - This task converts the input formats of the VTK enabled class into the individual formats of the components and vice versa if required.
  - This task can be implemented in a separate process object as well.
Application: Therm-OSS

Integration of modules/tools as Process Objects:

Data Transforming:
- This task converts the input formats of the VTK enabled class into the individual formats of the components and vice versa if required.
- This task can be implemented in a separate process object as well.

Pythonising: means that the Python interpreter can communicate with the component, which is necessary in first two cases of VTKising.
- using a shell command to execute a complete tool (Python build-in functions),
- using a wrapper generator to build C, C++ or Fortran to Python bindings,(swig, Boost:Python and pyfort)
- using communication protocols (e.g. Sockets, SOAP, Corba) for remote execution

Data Objects:
- VTK includes data objects for grid based datas (structured, unstructured grids, polygonal data).
- Therm-OSS requires the handling of other datas (e.g. geometry, mission definition, results).

Implementation:
- VTKising - map the information onto the existing VTK data objects
  Benefit: Reuse of existing objects, Deficit: Mapping is difficult/uncomplete/impossible
- Pythonising - new data objects derived from utk base classes
  Benefit: No restrictions, Deficit: No available process objects..

Information:
- Free definition - defining new data objects with properties/attributes
  Benefit: Enables quick prototyping, Deficit: Probably not general enough
- STEP definition - taking STEP TAS/NRF schemas as basis for the properties/attributes
  Benefit: Sophisticated information model, use of existing DEFicit: Complex model and implementation,
  except use of pyExpress (ESA/ESTEC)
Application: Therm-OSS

UML class diagram of STEP TAS/NRF observable item
- generate by Umbrello http://uml.sourceforge.net
- From pyExpress generated Python classes

Application: Therm-OSS

pyExpress extended with utk (Pythonizing)
- Extensions by deriving ExpressEntity from utkDataObject
- Add of 'capitalized' Methods
- Modification of ifls documentation classes
- Automatic support of the GUI and documentation environment
- Documentation important for interactive use
  - UML, DoxyGen

Collaboration diagramm generate by doxygen
Application: Therm-OSS

Simple Model build from STEP-TAS primitives.
- Cone
- Cylinder
- Disc
- Sphere
- Paraboloid
- Quadrilateral

Visualization of the primitives with VTK

Interactive editing
Application: Therm-OSS

- Simple Model build from STEP-TAS primitives.
- Tree representation in the ifs GUI.

Functional modules:
- Use case of the complete design and analysis cycle.
- Current work: development in the solver region.
  - FDM matrices for the primitives
  - Load vectors and matrices
  - Building the system equations
  - Solution
Concluding remarks

- Powerful tools for design, analysis and tool integration are available as OSS
- Open Source approach of a tool integration platform ifl is successful ... and makes fun
- Finding and evaluation of OSS is not easy
- License problem (free for research, not for commercial)
- Codes for the lumped parameter approach are hard to find
- Need for data and tool standards

Outlook for Therm-OSS

- First use case implementation finished in 2003
- Integration of alternative tools in 2004
- Common Component Architecture for High-Performance-Applications as one standard (?)