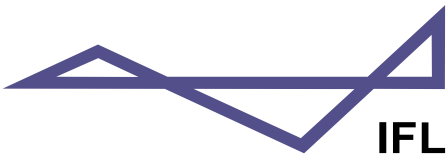
Capabilities of the Therm-OSS Tool

18th European Thermal & ECLS Software Workshop
ESA/ESTEC, Noordwijk 5-6 October 2004



Institut für Flugzeugbau und Leichtbau
Technische Universität Braunschweig

m.haupt@tu-bs.de

Matthias C. Haupt
Reinhold Niesner

Reinhard Schlitt (OHB)
Frank Bodendieck (OHB)

Charles Stroom (ESA/ESTEC)

Objectives of the Therm-OSS Project

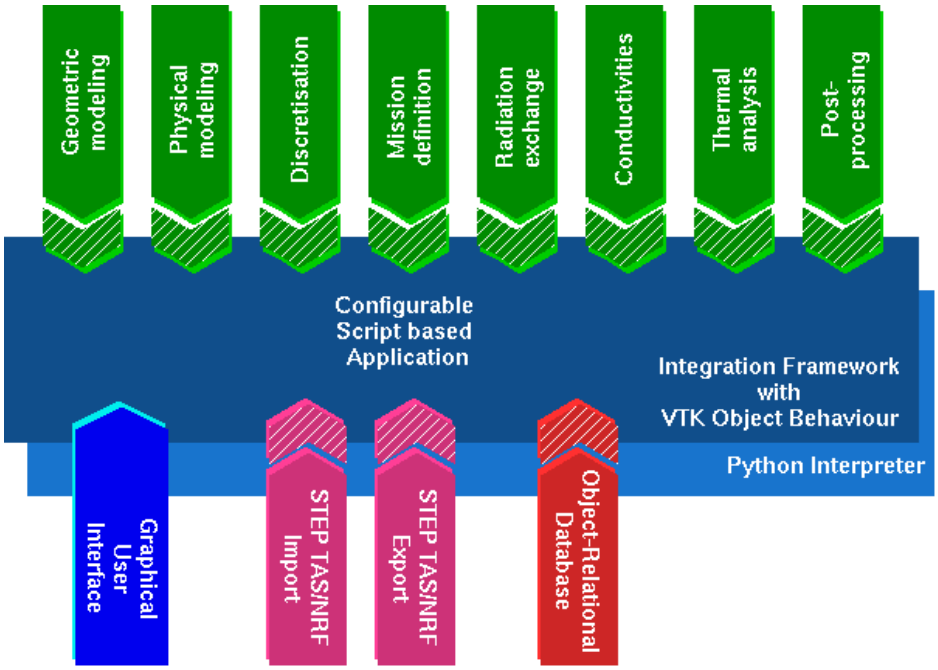
- To assess how OSS can be used to build applications
- To provide to developers a useful source of reference for their developments
- To assess whether the OSS approach could be useful as a distributed model for end-users

Introduction

- 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 (Primitives)
      - the environment
      - the mission and scenario
    - the definition and execution of the analysis (lumped parameter approach)
    - the evaluation or assessment of the results
    - Use of Open Source Software (OSS) as far as possible
  - Approach *in my role as a developer*
    - Survey of suitable OSS
    - Development of the general architecture
    - Implementation, test and ...

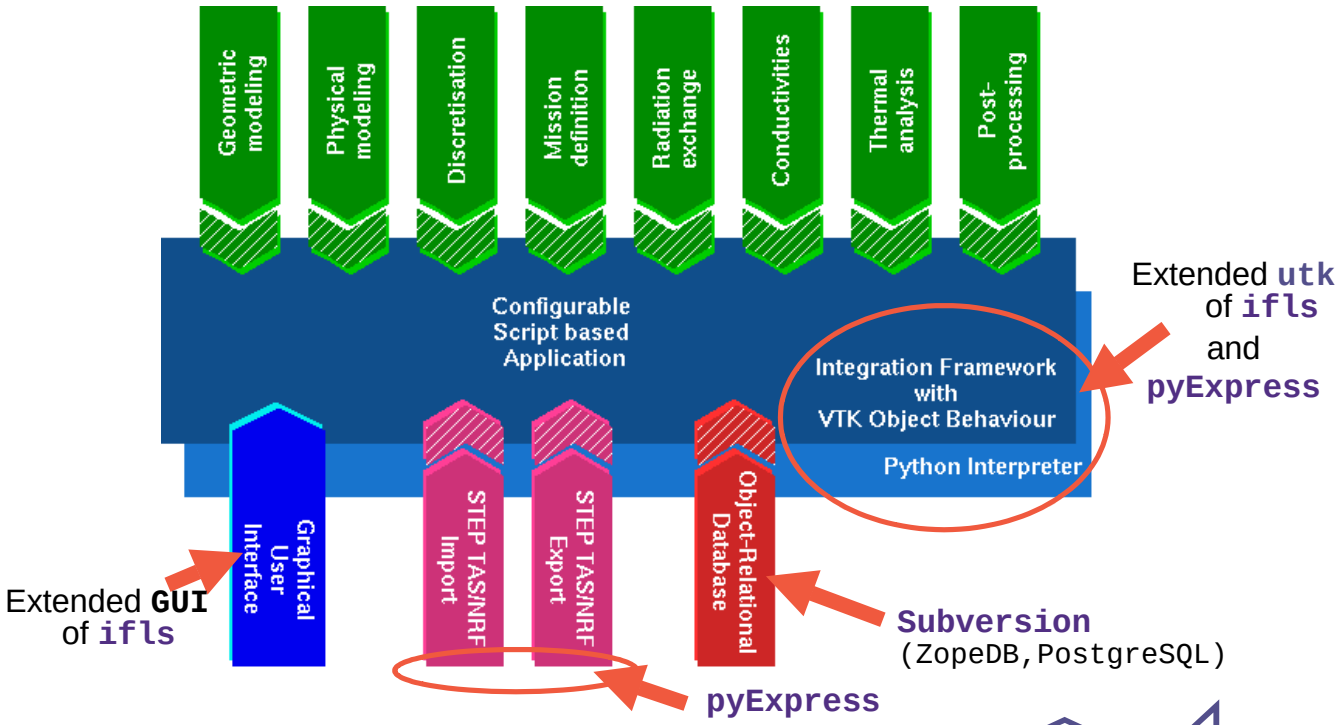
Architecture

- Proposed Design



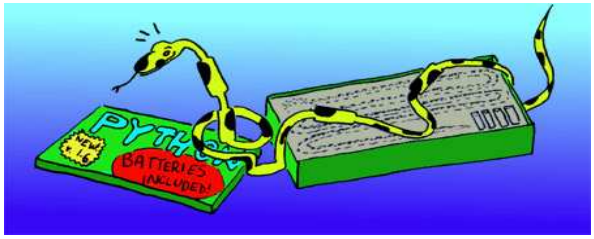
Architecture

- Engineering Infrastructure



Architectural Components

- **ifls** uses for Implementation and Scripting
  - **Python**
    Object-orientiented 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/numpy**:
  Vector / matrix objects
- **Scientific Python**:
  Scientific tools, MPI, NetCDF, Optimization, ...

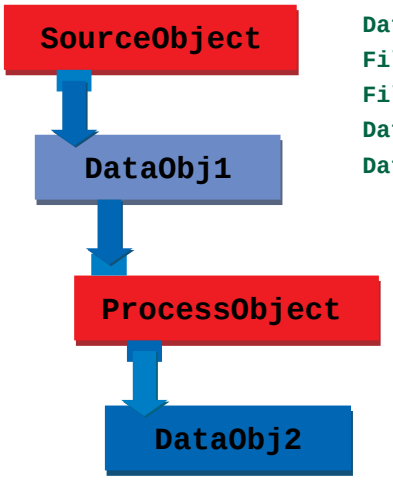
Interface generators

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

Architectural Components

- **ifls** extents the **vtk** pipeline

- DataObjects
  represent information
- ProcessObjects
  operates on input data to generate output data
- Pipeline Execution
  causes processObjects to operate



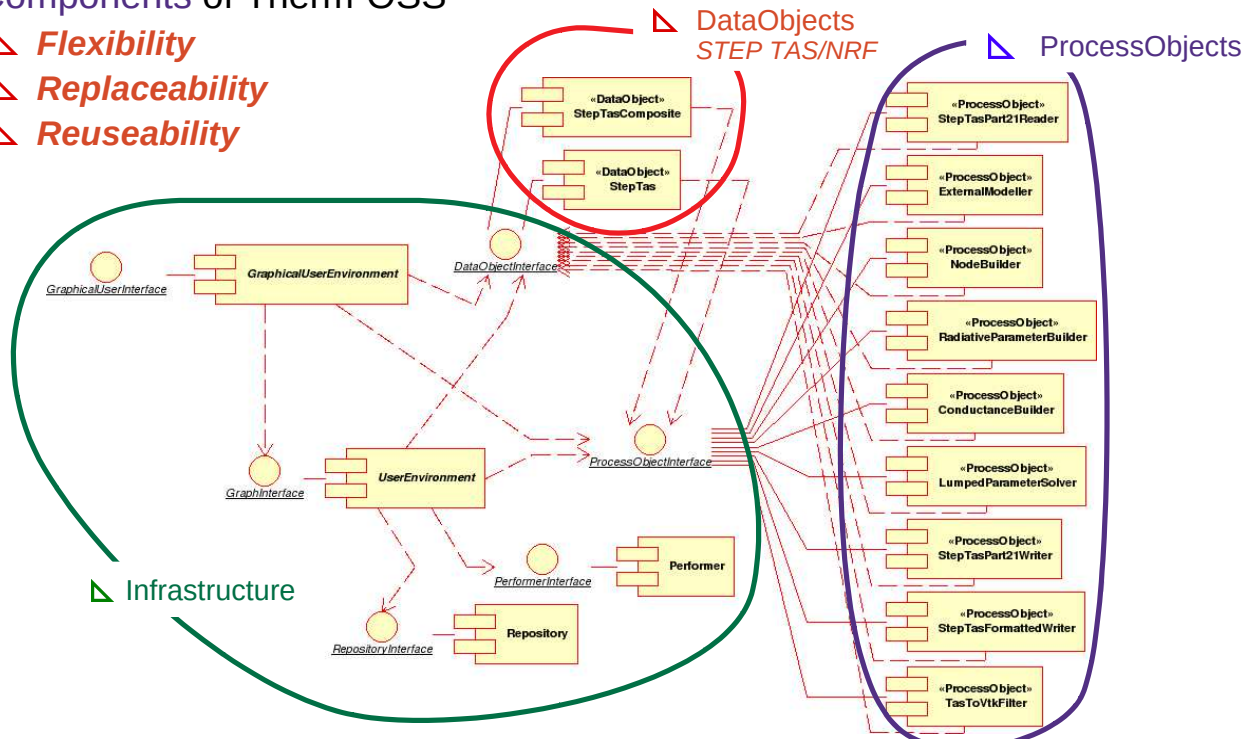

```
Reader = SourceObject()
DataObj1 = Reader.GetOutput()
Filter = ProcessObject()
Filter.SetInput( DataObj1 )
DataObj2 = Filter.GetOutput()
DataObj2.Update()
```

New approach of solving problems with software components

utk connects **vtk** and pure **Python** classes to maintain pipeline mechanisms

Architectural Components

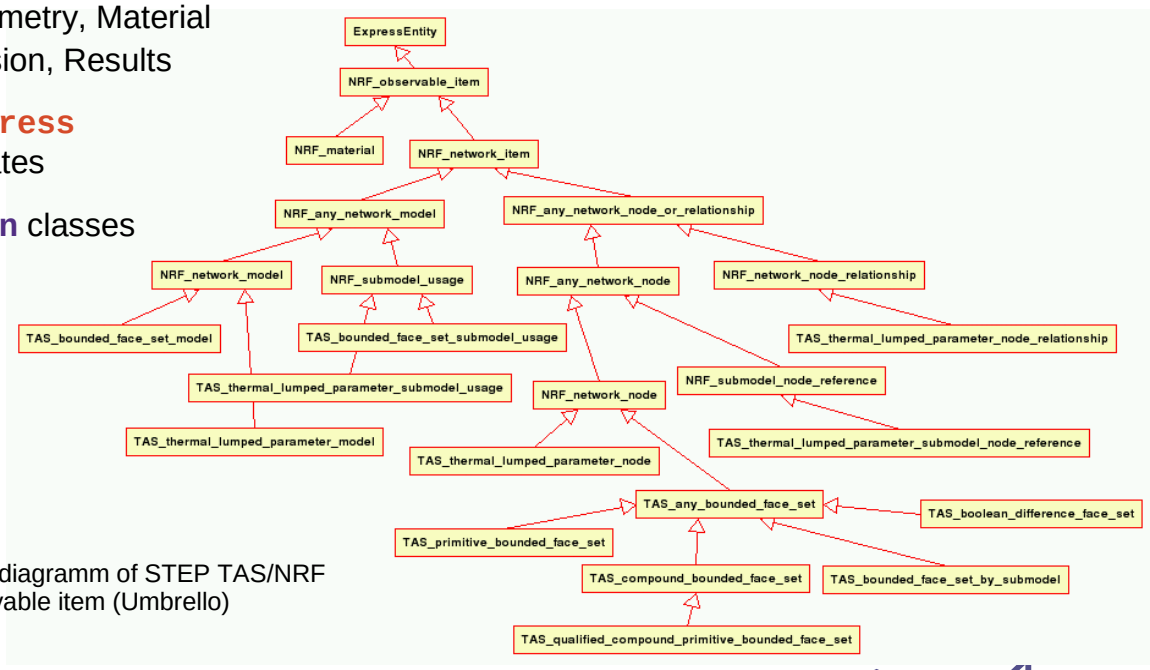
- Components of Therm-OSS
  - ***Flexibility***
  - ***Replaceability***
  - ***Reuseability***



Architectural Components

- STEP TAS/NRF integration
  - STEP TAS/NRF data model
    - Geometry, Material
    - Mission, Results
  - **PyExpress** generates

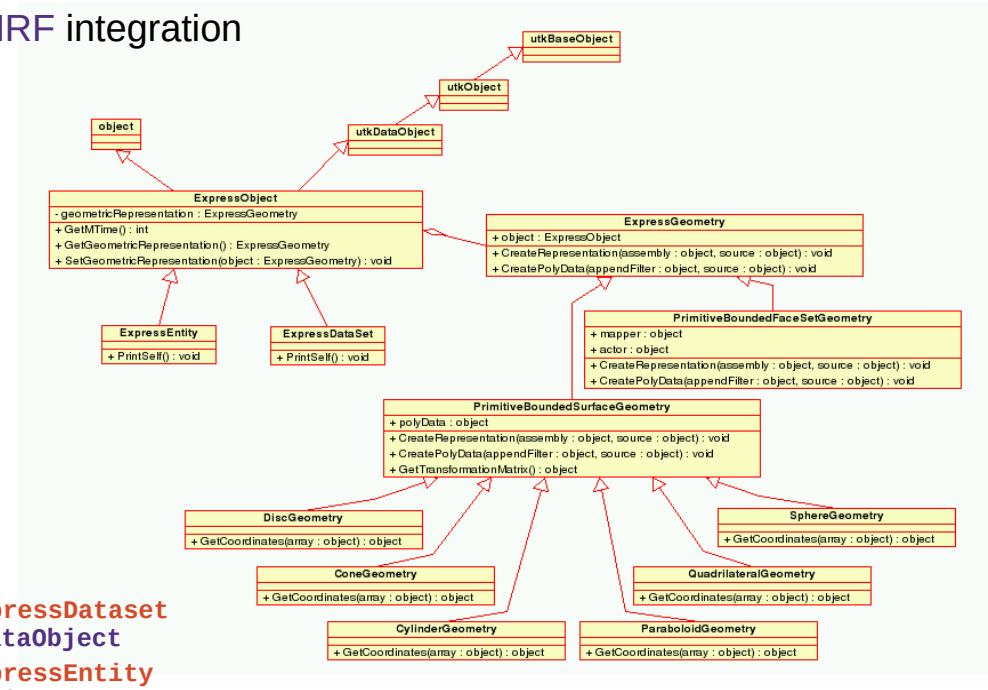
    Python classes



Class diagramm of STEP TAS/NRF observable item (Umbrello)

Architectural Components

- STEP TAS/NRF integration

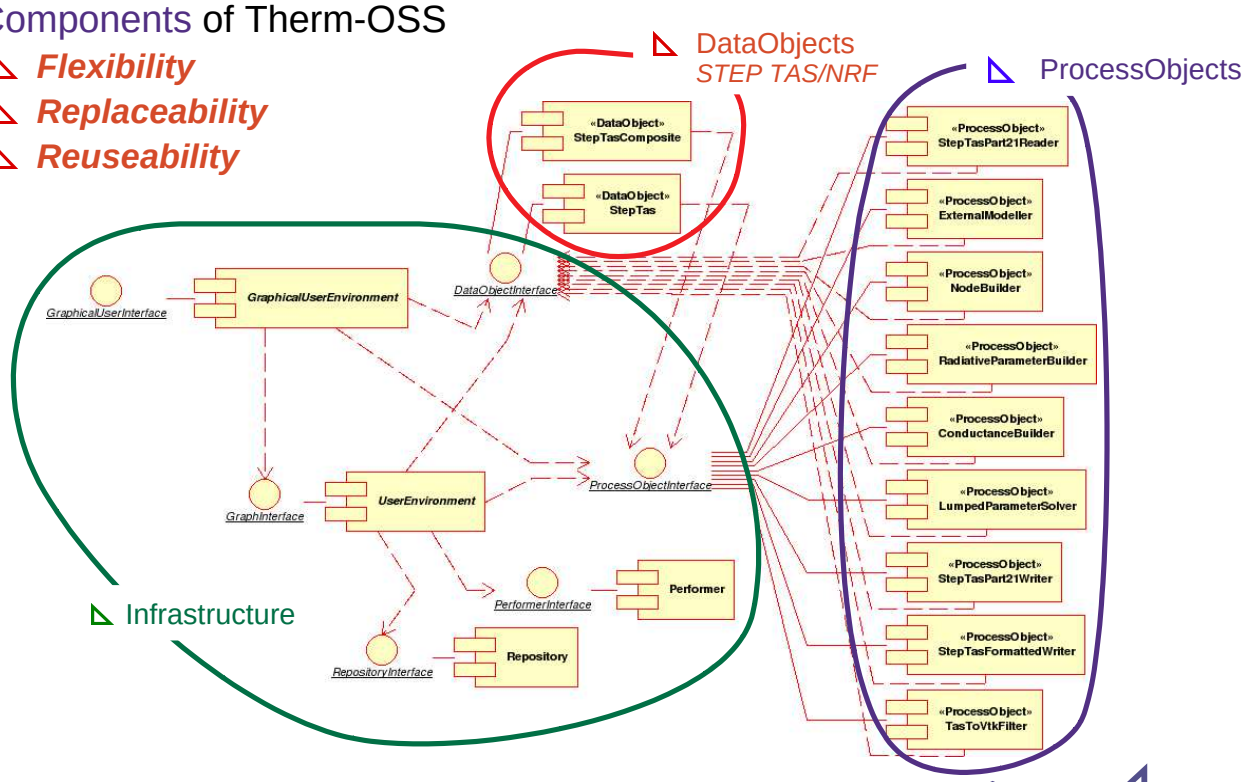


- Inheritance
  - Deriving **ExpressDataset** from **utkDataObject**
  - Deriving **ExpressEntity** from **utkObject**
- Adding attribute **model**
  - Representation / physics

Class diagramm of STEP TAS/NRF UTK integration

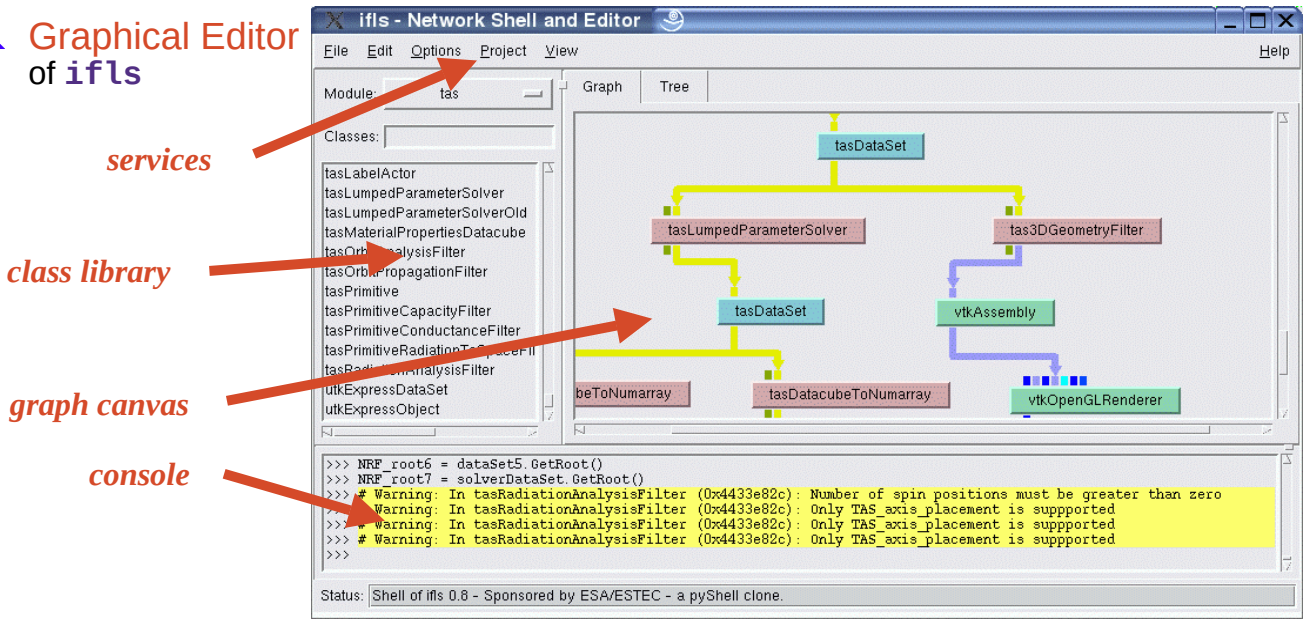
Architectural Components

- Components of Therm-OSS
  - ***Flexibility***
  - ***Replaceability***
  - ***Reuseability***



Architectural Components

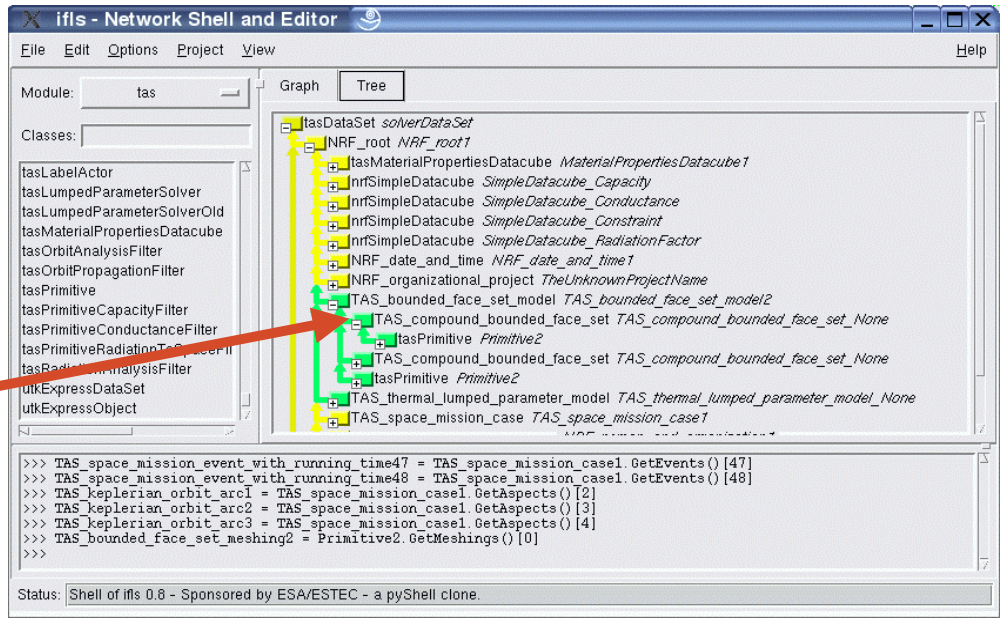
- Graphical Editor of **`ifls`**



- Interactive manipulation and visual programming of networks/dataflow
- Analyses and visualizes the programmed object interactions/networks (tree / **graph**)
- **`Python`** codes are executable without the graphical editor in batch mode

Architectural Components

- Graphical Editor of **`ifls`**

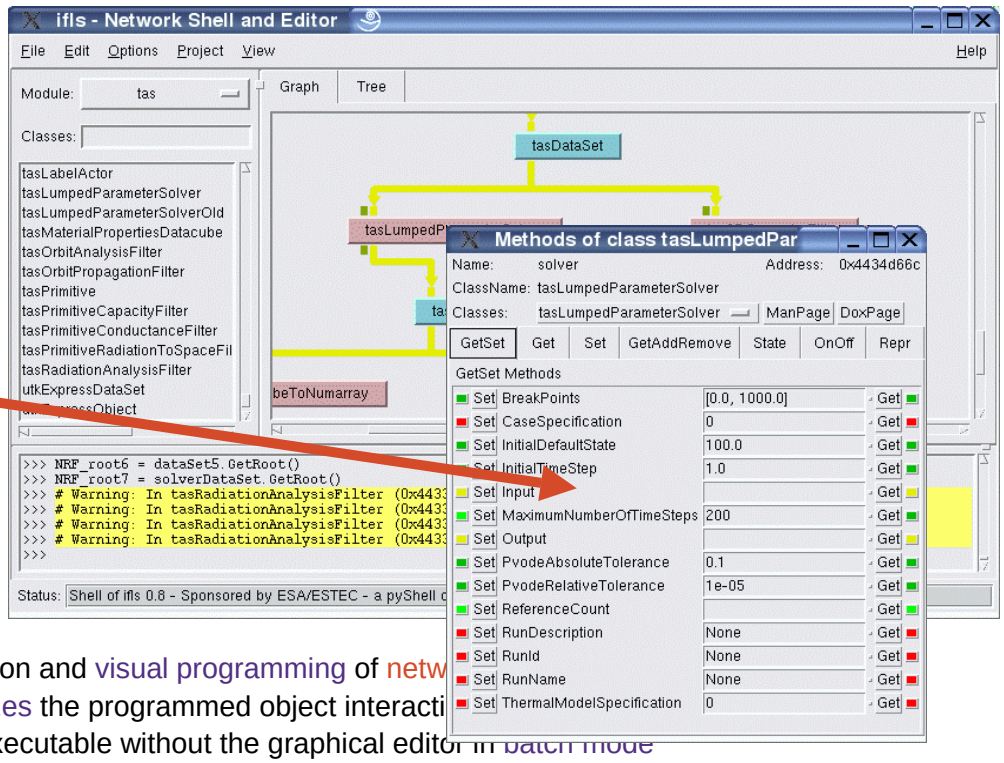


tree canvas

- Interactive manipulation and visual programming of networks/dataflow
- Analyses and visualizes the programmed object interactions/networks (**tree** / graph)
- **`Python`** codes are executable without the graphical editor in batch mode

Architectural Components

- Graphical Editor of **`ifls`**

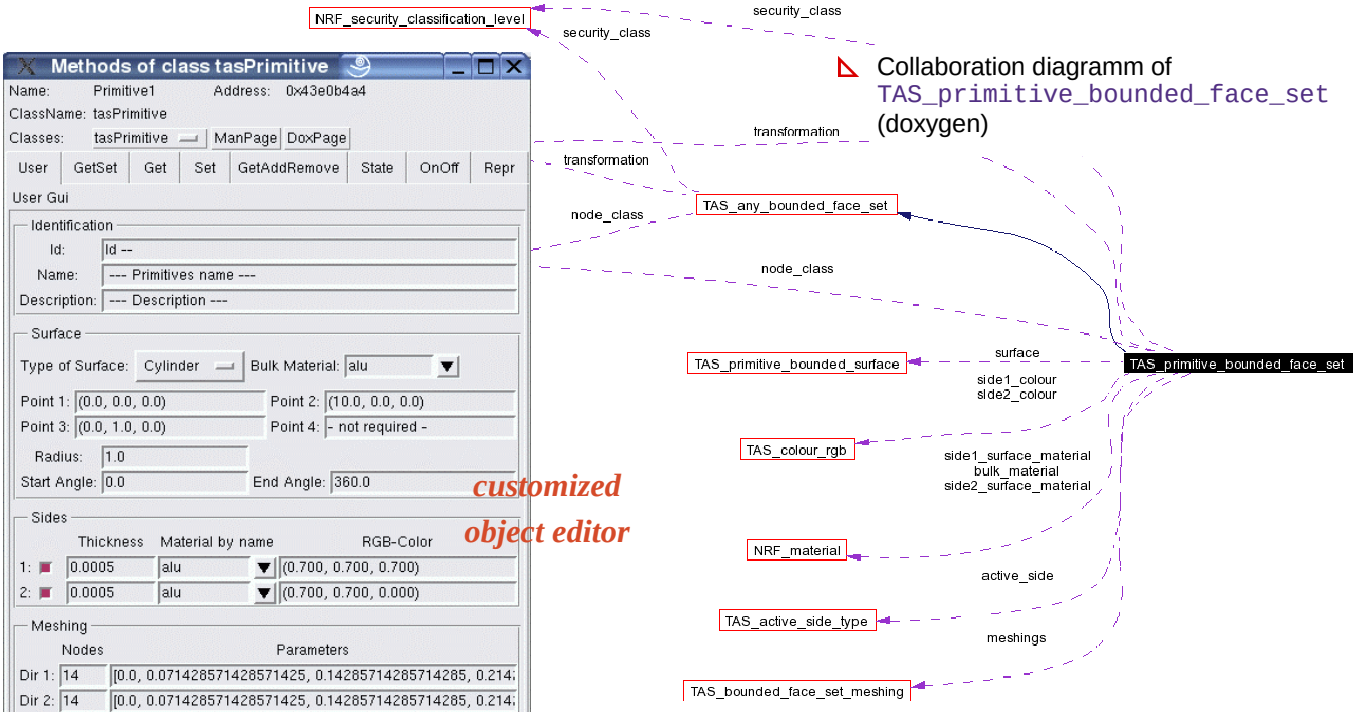


generic object editor

- Interactive manipulation and visual programming of netw
- Analyses and visualizes the programmed object interacti
- **`Python`** codes are executable without the graphical editor in batch mode

Architectural Components

- Composite Classes ... to hide the aggregation ... for easy use.



customized object editor

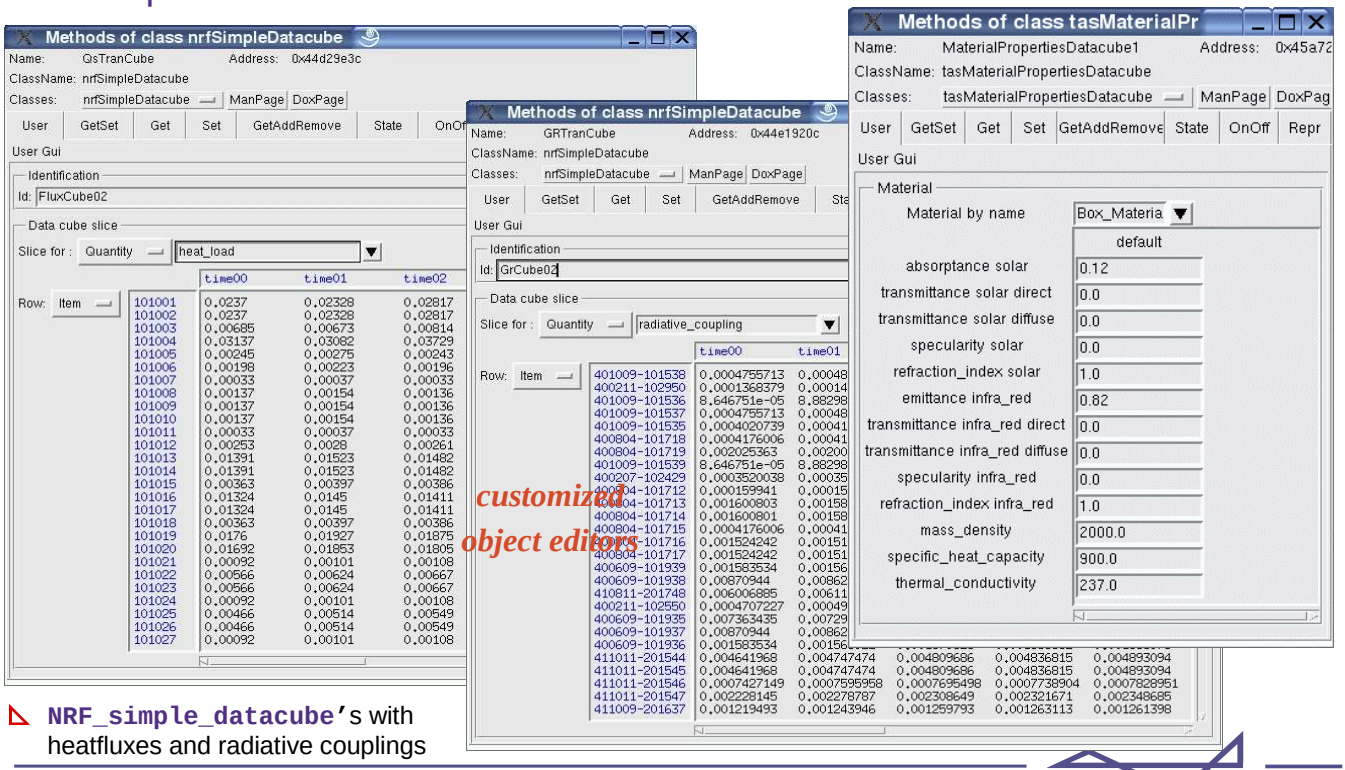
- Collaboration diagramm of `TAS_primitive_bounded_face_set` (doxygen)

IFL

Architectural Components

- Composite Classes

- `TAS_material_properties_datacube`



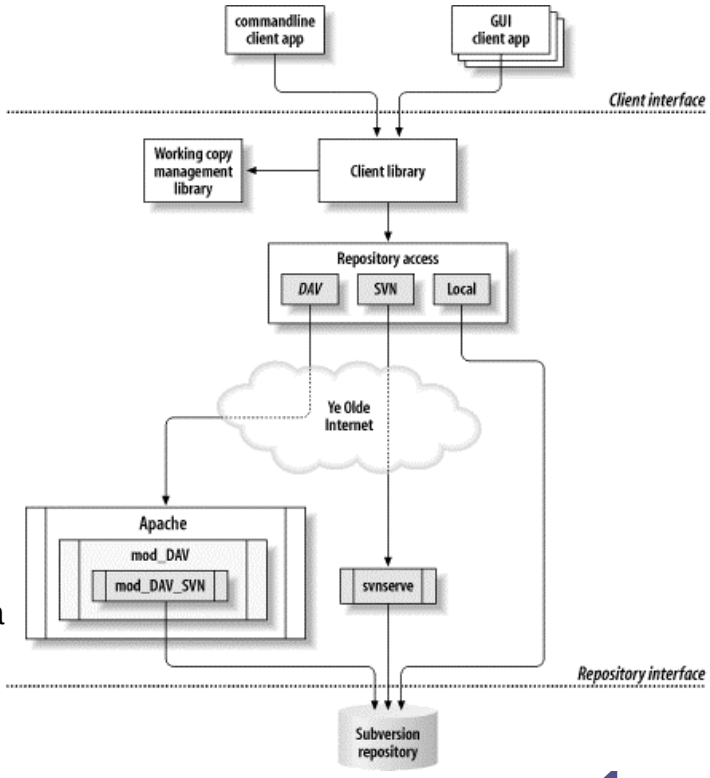
customized object editors

- `NRF_simple_datacube`'s with heatfluxes and radiative couplings

IFL

Architectural Components

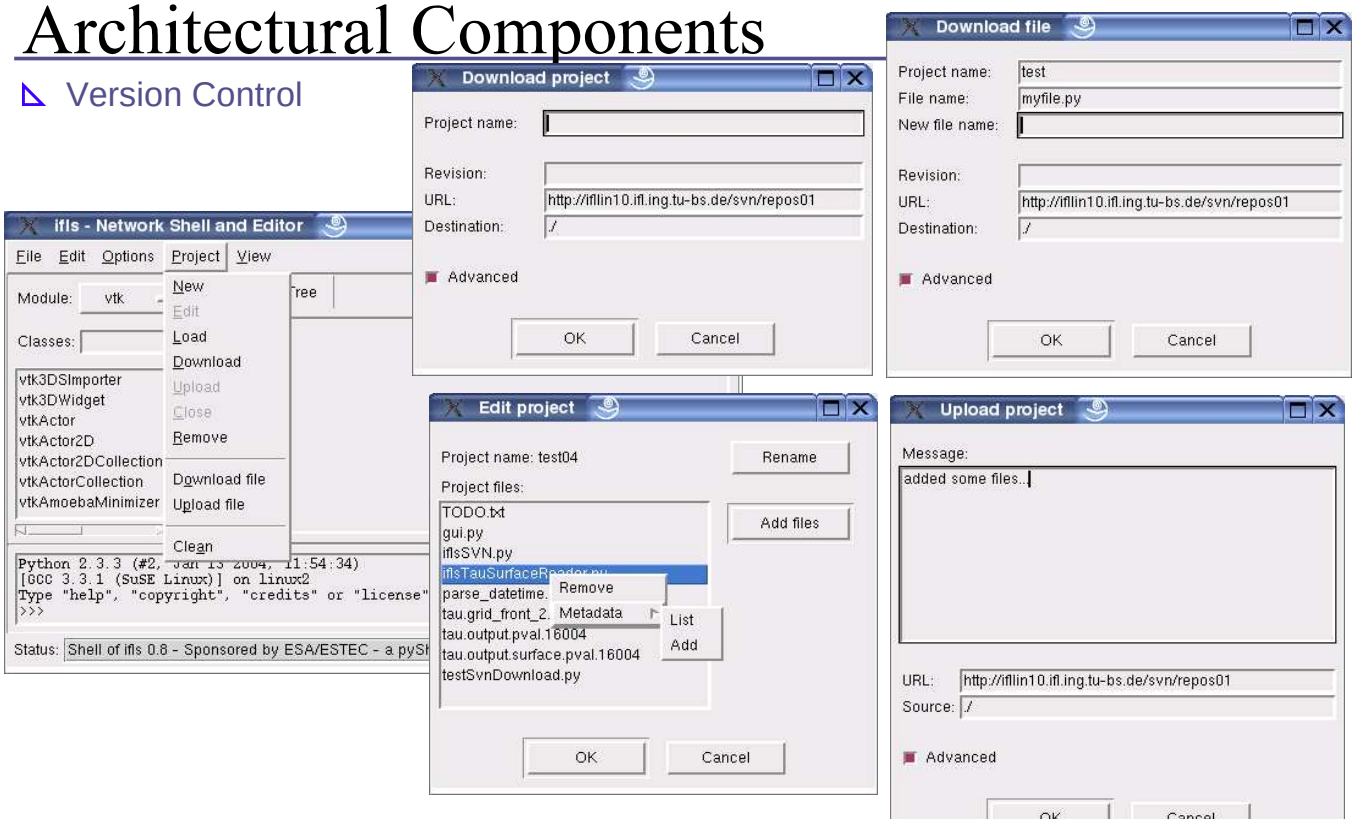
- Version Control
  - File oriented environment
    - Python scripts, STEP-TAS files, …
  - Subversion
    - Manages Files (incl. binary)
      … and Directories
    - Central repository
      … similar to CVS
    - Choice of network layer
      … (http/svn/ssh)
    - Collection of shared libraries
      … Python bindings
  - Extension of the environment with a *simple* client



IFL

Architectural Components

- Version Control

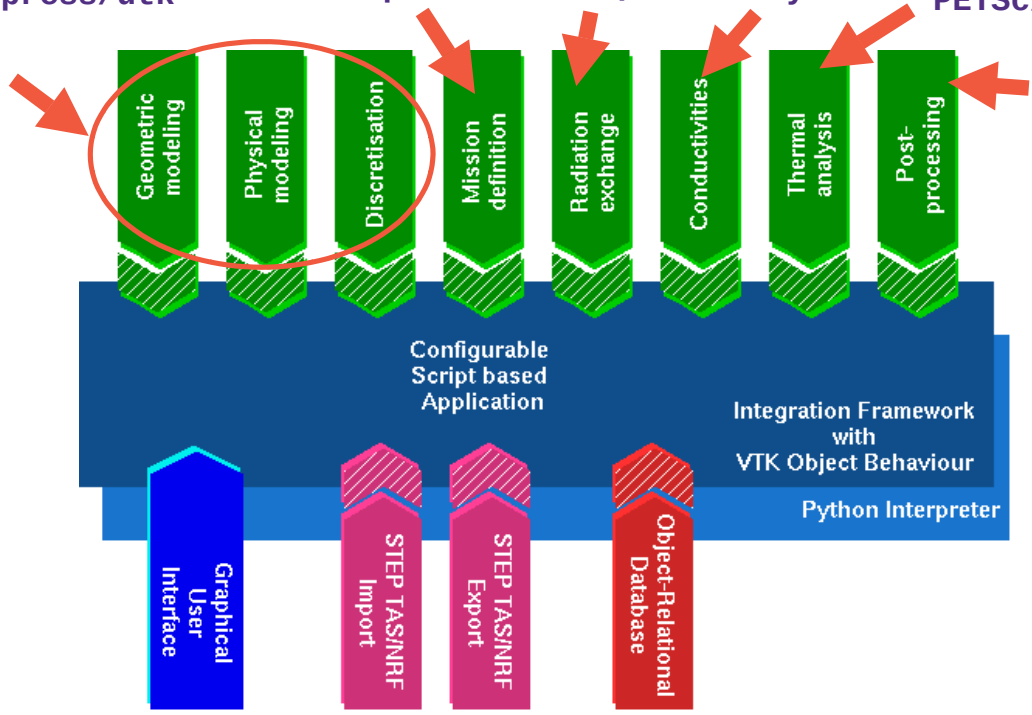


- Extension of the environment with a *simple* client

IFL

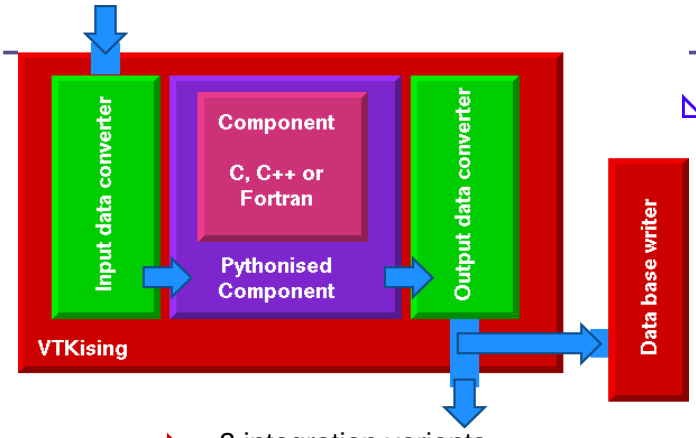
Architecture

pyExpress/utk
K3D
(Predict) Topic
(Renderpark) Topic
Python
PETSc/Sundials
VTK
Gnuplot
(Chaco)

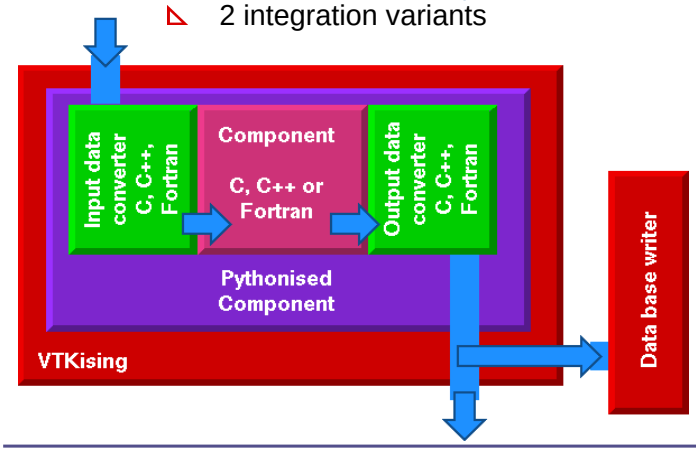


- Functional modules

Integration



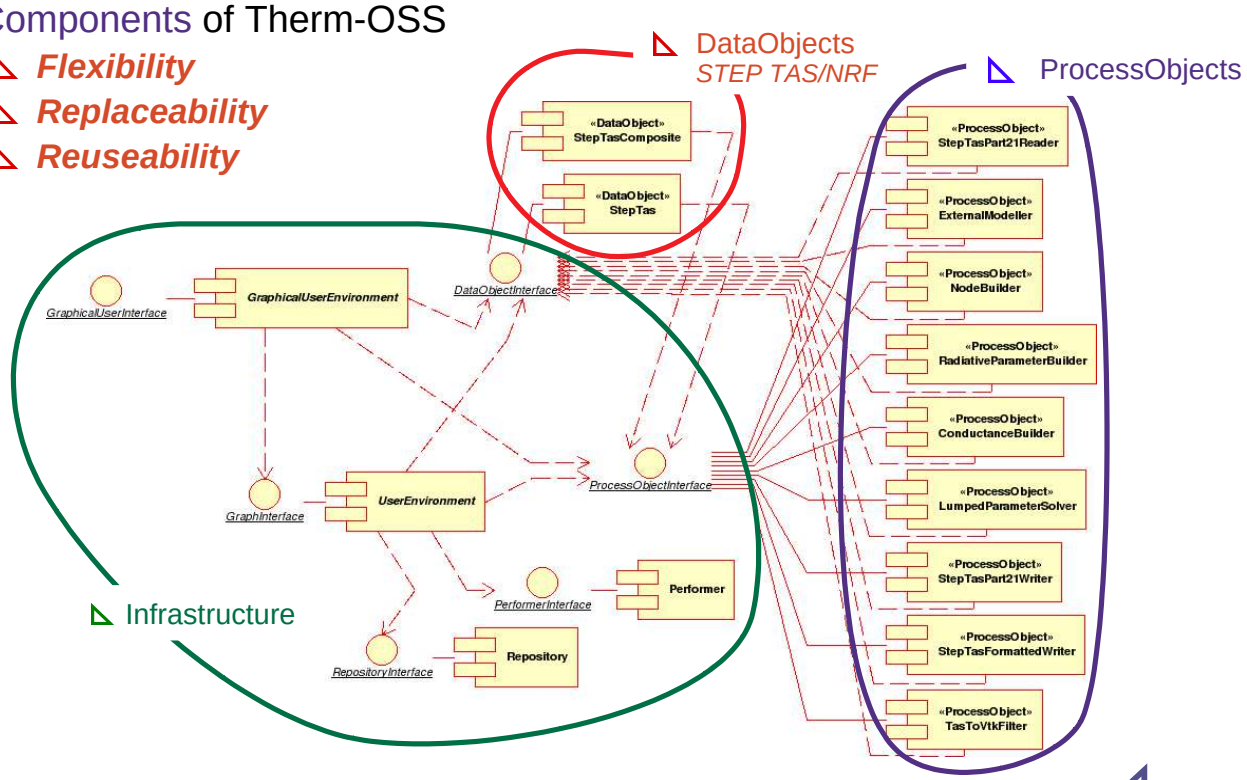
- 2 integration variants



- Integration of modules / tools as ProcessObjects:
  - **VTK** ising makes tool behave like a ProcessObject.
  - Data Transforming converts the input format into the tools format and vice versa *if required.*
  - **Python**-ising tool must talk **Python**.

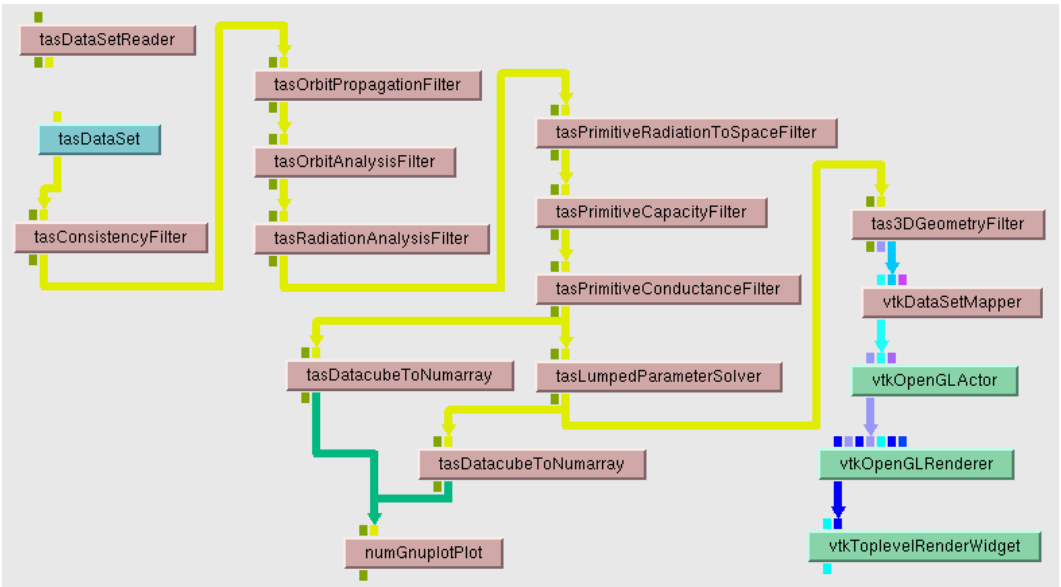
Architectural Components

- Components of Therm-OSS
  - ***Flexibility***
  - ***Replaceability***
  - ***Reuseability***



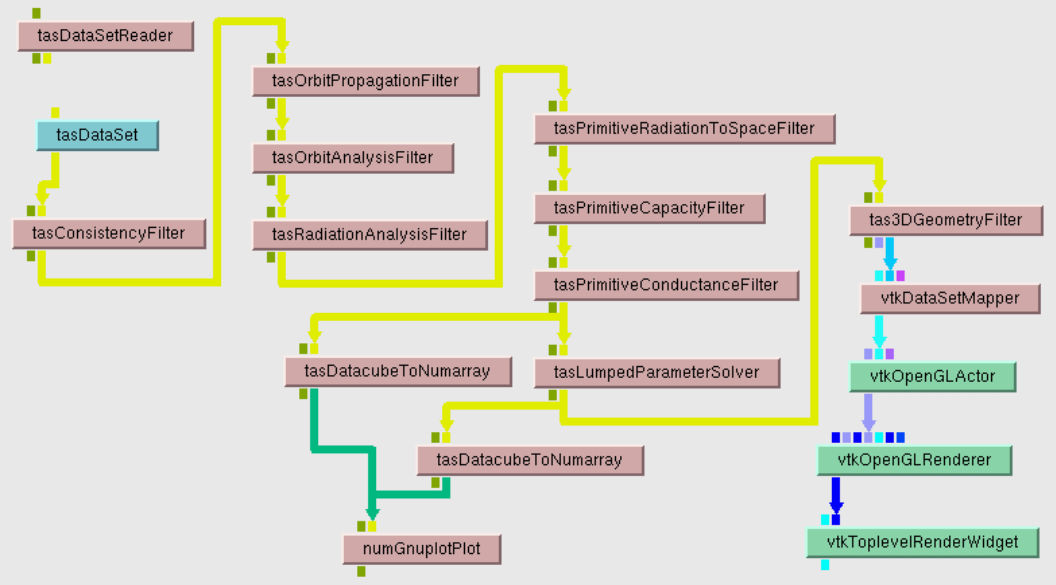
Application

- The Network ... from persistence storage, script or via GUI



Application

- The Network ... from persistence storage, script or via GUI



Build your own application

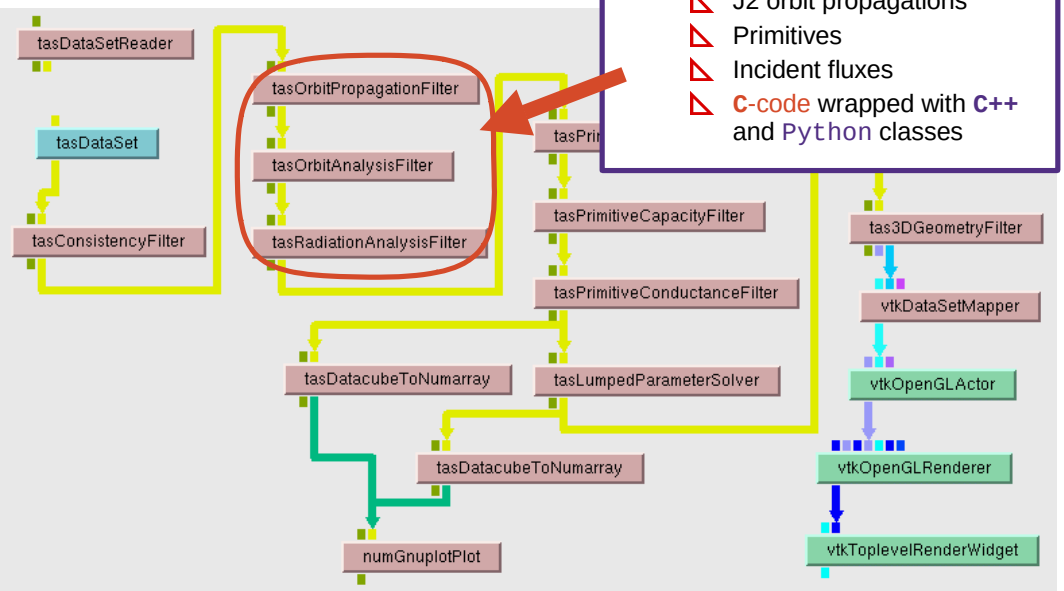
Include your own components

IFL

Application

- The Network

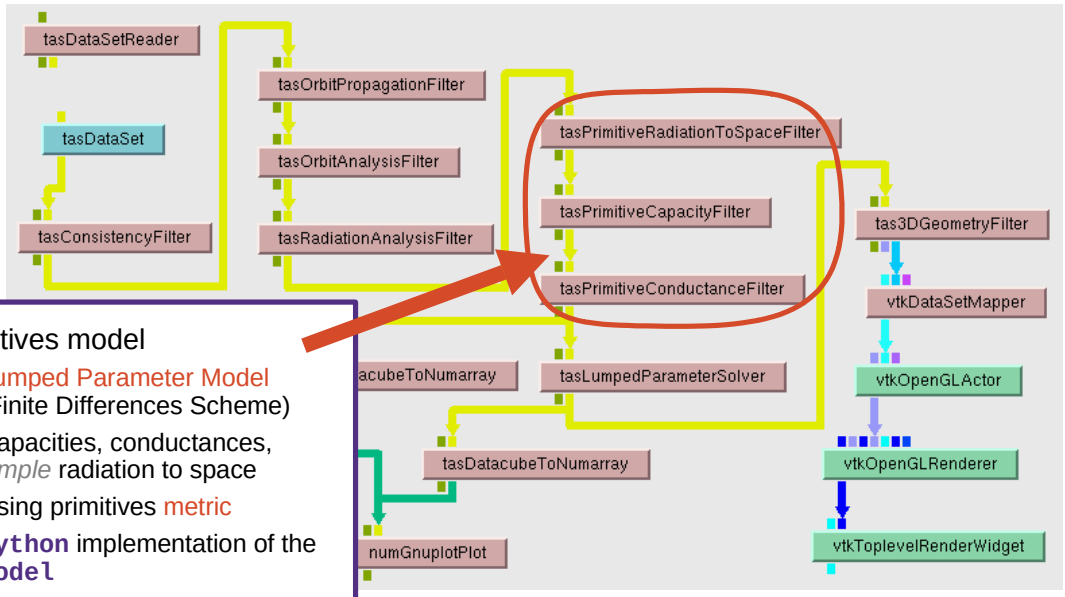
- Topic (ESA/ESTEC):
  - Earth orbits
  - J2 orbit propagations
  - Primitives
  - Incident fluxes
  - C-code wrapped with C++ and Python classes



IFL

Application

- The Network

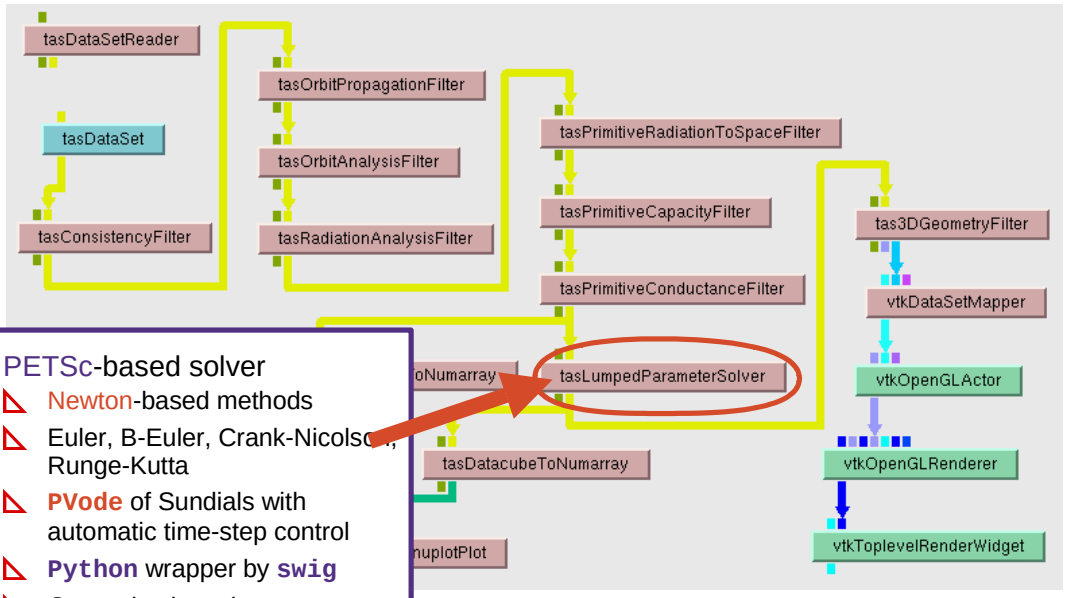


- Primitives model
  - Lumped Parameter Model (Finite Differences Scheme)
  - Capacities, conductances, *simple* radiation to space
  - Using primitives metric
  - **`Python`** implementation of the **`model`**

IFL

Application

- The Network



- PETSc-based solver
  - Newton-based methods
  - Euler, B-Euler, Crank-Nicolson, Runge-Kutta
  - **`PVode`** of Sundials with automatic time-step control
  - **`Python`** wrapper by **`swig`**
  - General solver class
  - **`Function`** and **`Jacobian`**
  - Robust and fast

IFL

Applic

The Netw

PE

OHB-Example

```
# Stdout: Nonlinear solver
# Stdout:                   Iteration   Residual
# Stdout: SNES-Convergence    0        1.38101e+02
# Stdout: SNES-Convergence    1        4.89064e-01
# Stdout: SNES-Convergence    2        4.98388e-04
# Stdout: SNES-Convergence    3        7.53985e-08
# Stdout: SNES ConvergedReason: SNES_CONVERGED_FNORM_RELATIVE (F < F_mintol*F_initial)
# Note: SNES Number of iterations: 3; CPU time: 3.653200
# Stdout: SNES Object:
# Stdout:   type: ls
# Stdout:     line search variant: SNESCubicLineSearch
# Stdout:     alpha=0.0001, maxstep=1e+08, steptol=1e-12
# Stdout:   maximum iterations=50, maximum function evaluations=10000
# Stdout:   tolerances: relative=1e-08, absolute=1e-50, solution=1e-08
# Stdout:   total number of linear solver iterations=18
# Stdout:   total number of function evaluations=4
# Stdout:   KSP Object:
# Stdout:     type: gmres
# Stdout:       GMRES: restart=30, using Classical (unmodified) Gram-Schmidt Orthogonalization with no it
# Stdout:       GMRES: happy breakdown tolerance 1e-30
# Stdout:     maximum iterations=10000, initial guess is zero
# Stdout:     tolerances:  relative=1e-05, absolute=1e-50, divergence=10000
# Stdout:     left preconditioning
# Stdout:   PC Object:
# Stdout:     type: ilu
# Stdout:       ILU: 10 levels of fill
# Stdout:       ILU: max fill ratio allocated 1
# Stdout:       ILU: tolerance for zero pivot 1e-12
# Stdout:            out-of-place factorization
# Stdout:            matrix ordering: nd
# Stdout:            Factored matrix follows
# Stdout:       Matrix Object:
# Stdout:         type=seqaij, rows=3138, cols=3138
# Stdout:         total: nonzeros=256010, allocated nonzeros=256010
# Stdout:           not using I-node routines
# Stdout:     linear system matrix = precond matrix:
# Stdout:     Matrix Object:
# Stdout:       type=seqaij, rows=3138, cols=3138
# Stdout:       total: nonzeros=21380, allocated nonzeros=56898
# Stdout:         not using I-node routines
```

IFL

Applic

The Netw

PE

OHB-Example

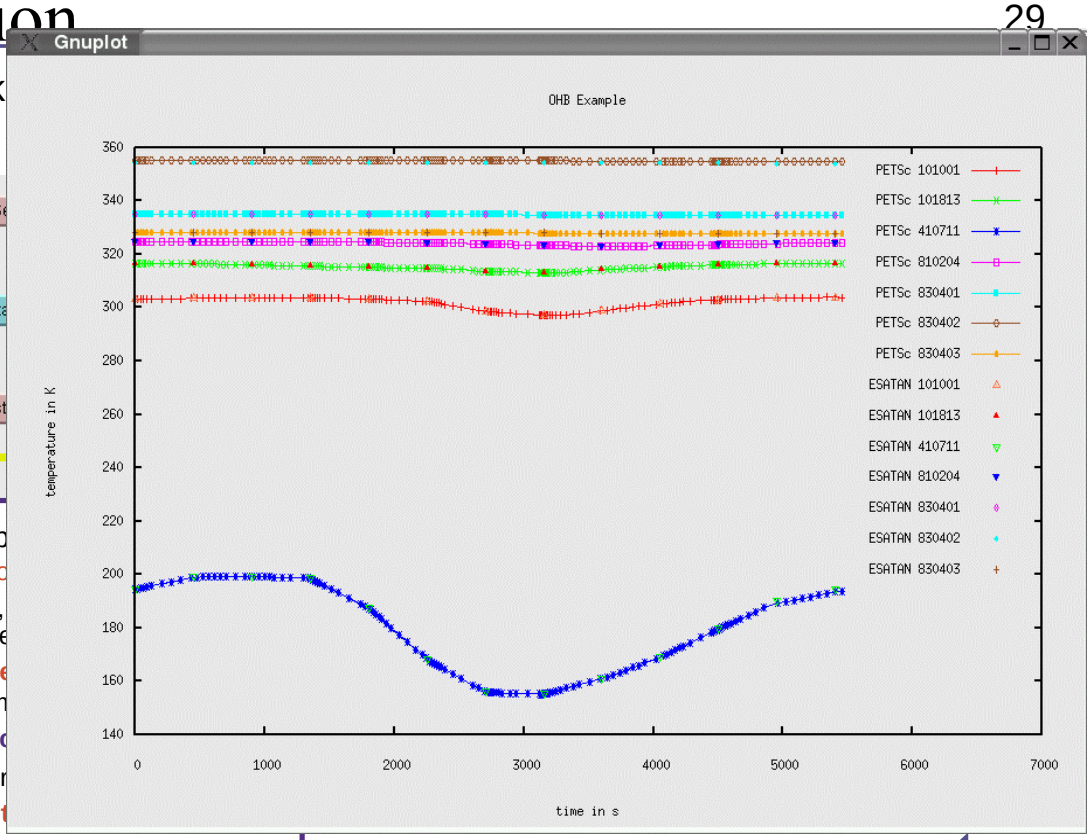
```
# Stdout: Transient Solver
# Stdout:          Timestep   Time
# Stdout: TS         0       0.00000e+00
# Stdout: TS         1       1.90181e-03
# Stdout: TS         2       1.90200e+01
   ....
# Stdout: TS       129       5.39841e+03
# Stdout: TS       130       5.45760e+03
# Note: TS Number of time steps: 130; final time: 5457.602388; CPU time: 34.318804
# Stdout: TS Object:
# Stdout:   type: pvode
# Stdout:   PVode integrater does not use SNES!
# Stdout:   PVode integrater type BDF: backward differentiation formula
# Stdout:   PVode abs tol 0.01 rel tol 1e-06
# Stdout:   PVode linear solver tolerance factor 0.05
# Stdout:   PVode GMRES max iterations (same as restart in PVODE) 5
# Stdout:   PVode using unmodified (classical) Gram-Schmidt for orthogonalization in GMRES
# Stdout:     PC Object:
# Stdout:       type: ilu
# Stdout:         ILU: 10 levels of fill
# Stdout:         ILU: max fill ratio allocated 1
# Stdout:         ILU: tolerance for zero pivot 1e-12
# Stdout:              out-of-place factorization
# Stdout:              matrix ordering: nd
# Stdout:              Factored matrix follows
# Stdout:         Matrix Object:
# Stdout:           type=seqaij, rows=3138, cols=3138
# Stdout:           total: nonzeros=256010, allocated nonzeros=256010
# Stdout:             not using I-node routines
# Stdout:       linear system matrix = precond matrix:
# Stdout:       Matrix Object:
# Stdout:         type=seqaij, rows=3138, cols=3138
# Stdout:         total: nonzeros=21380, allocated nonzeros=56898
# Stdout:           not using I-node routines
# Stdout:   maximum steps=1000
# Stdout:   maximum time=5400
# Stdout:   total number of nonlinear solver iterations=212
# Stdout:   total number of linear solver iterations=197
```

Robust and fast

IFL

Application

- The Network

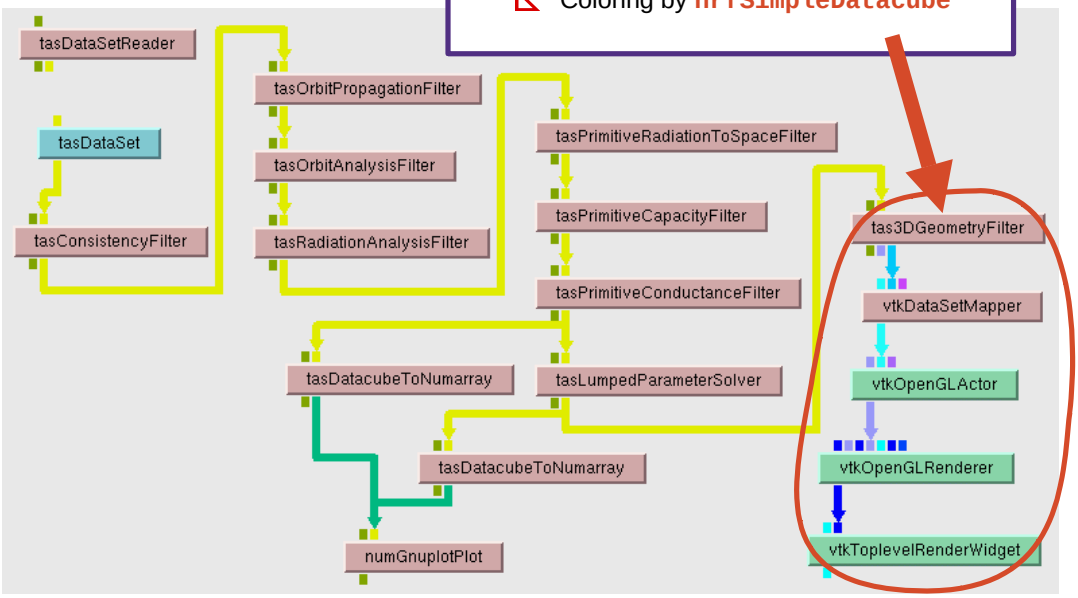


- PETSc-b
  - Newto
  - Euler, Runge
  - PVode autom
  - Pytho
  - Gener
  - Funct
  - Robust and fast

Application

- The Network

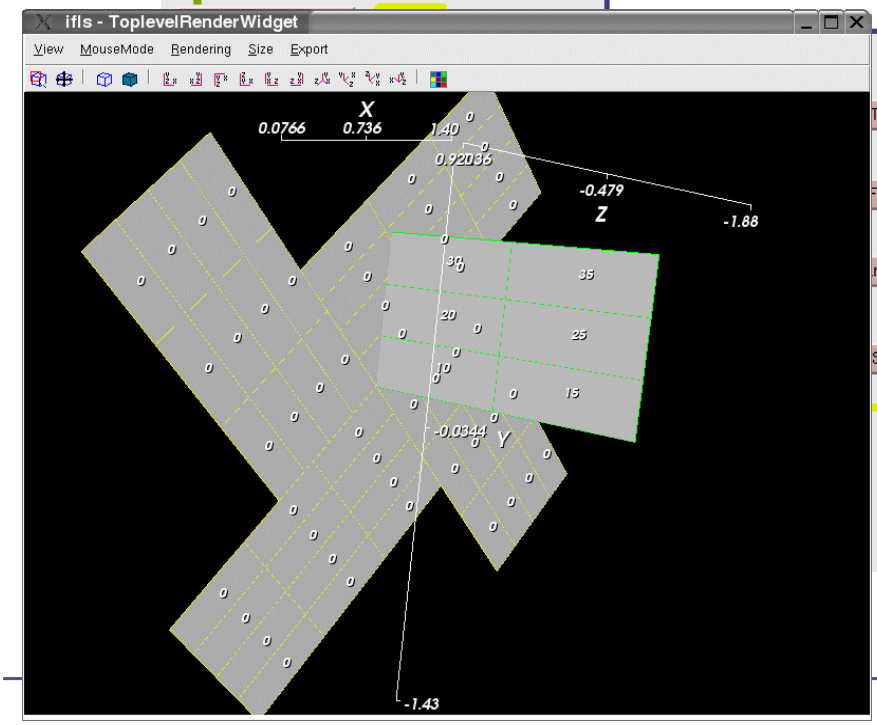
- vtk-based visualization
  - STEP TAS to vtk conversion
  - vtk-processing and rendering
  - 3D-geometry incl. Labels
  - Coloring by nrfSimpleDatacube

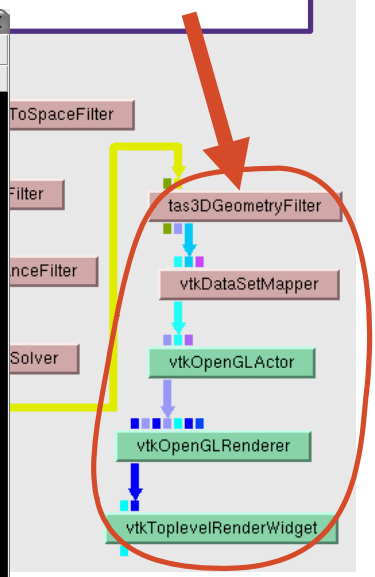


Application

- The Network

- **vtk**-based visualization
  - **STEP TAS** to **vtk** conversion
  - **vtk**-processing and rendering
  - 3D-geometry incl. Labels
  - Coloring by **nrfSimpleDatacube**

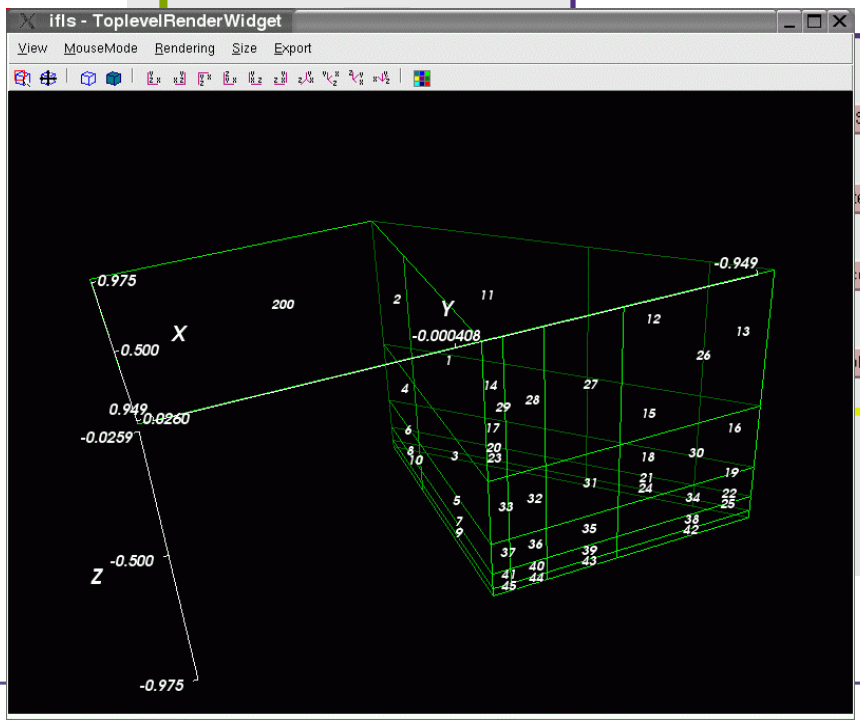


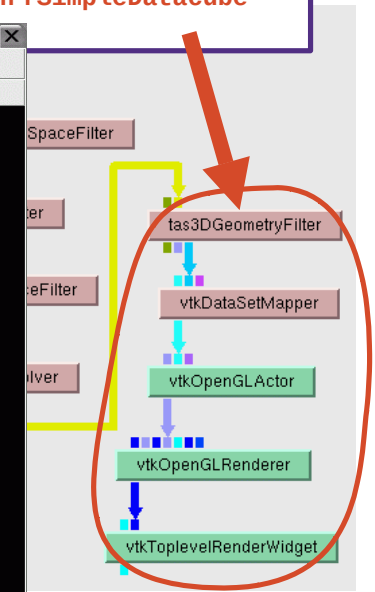


Application

- The Network

- **vtk**-based visualization
  - **STEP TAS** to **vtk** conversion
  - **vtk**-processing and rendering
  - 3D-geometry incl. Labels
  - Coloring by **nrfSimpleDatacube**

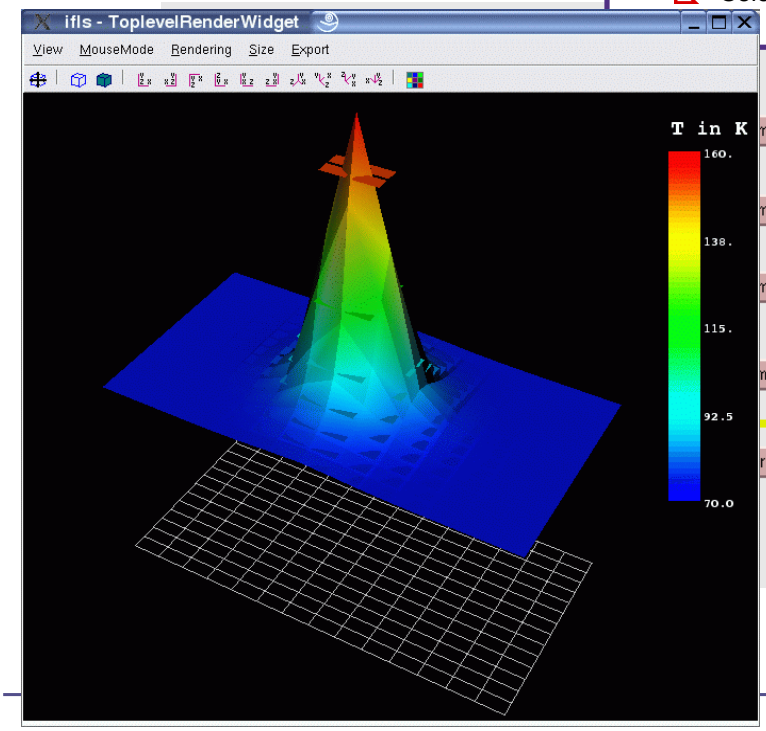




Application

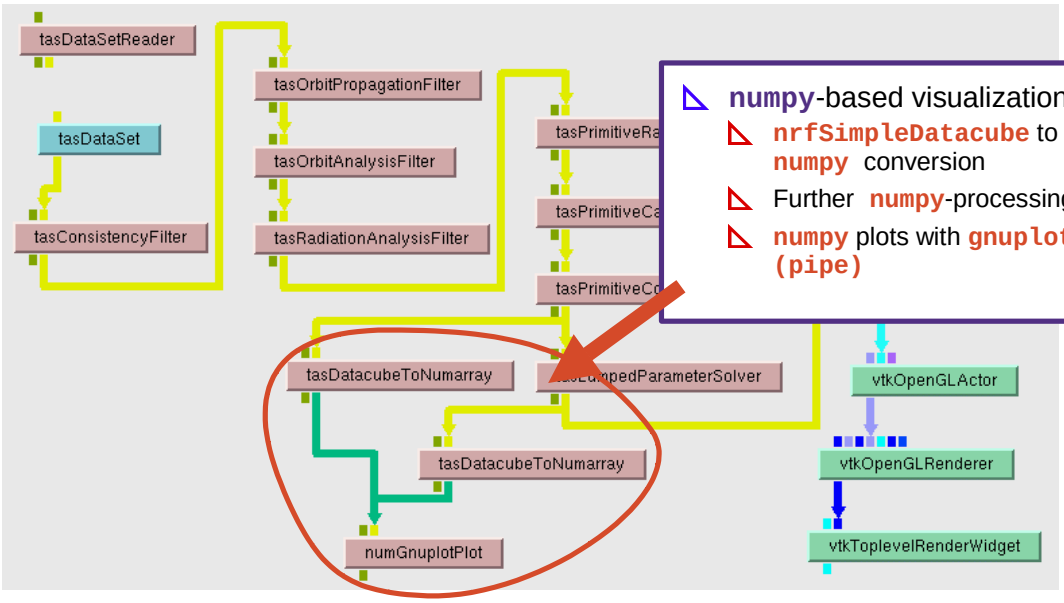
- The Network

- **vtk**-based visualization
  - **STEP TAS** to **vtk** conversion
  - **vtk**-processing and rendering
  - 3D-geometry incl. Labels
  - Coloring by **nrfSimpleDatacube**



Application

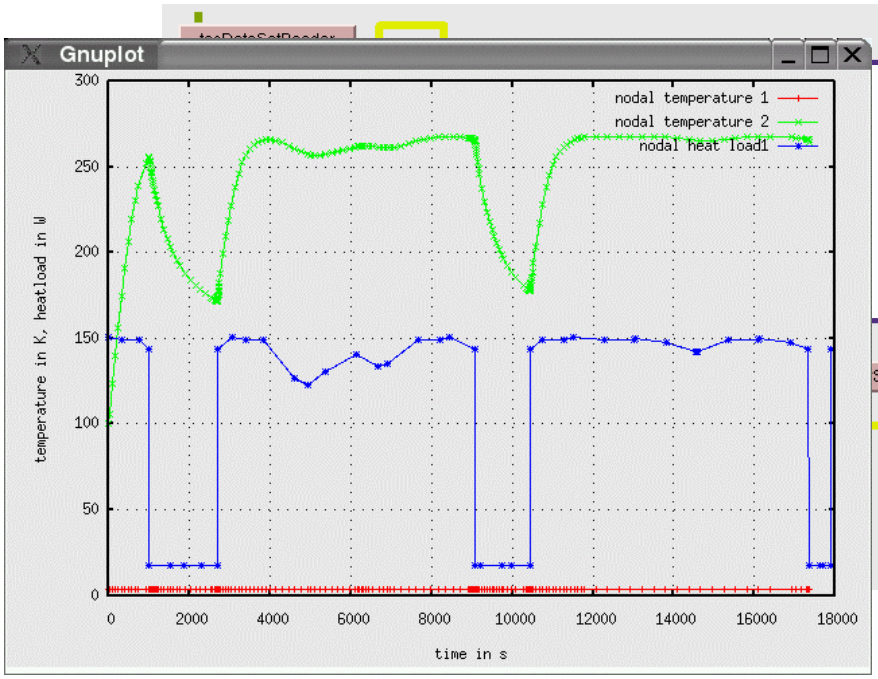
- The Network



- **numpy**-based visualization
  - **nrfSimpleDatacube** to **numpy** conversion
  - Further **numpy**-processing
  - **numpy** plots with **gnuplot (pipe)**

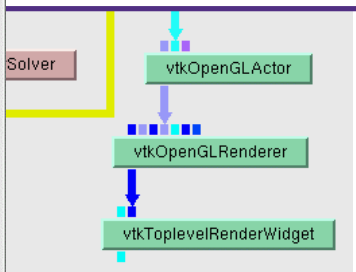
Application

- The Network



numpy-based visualization

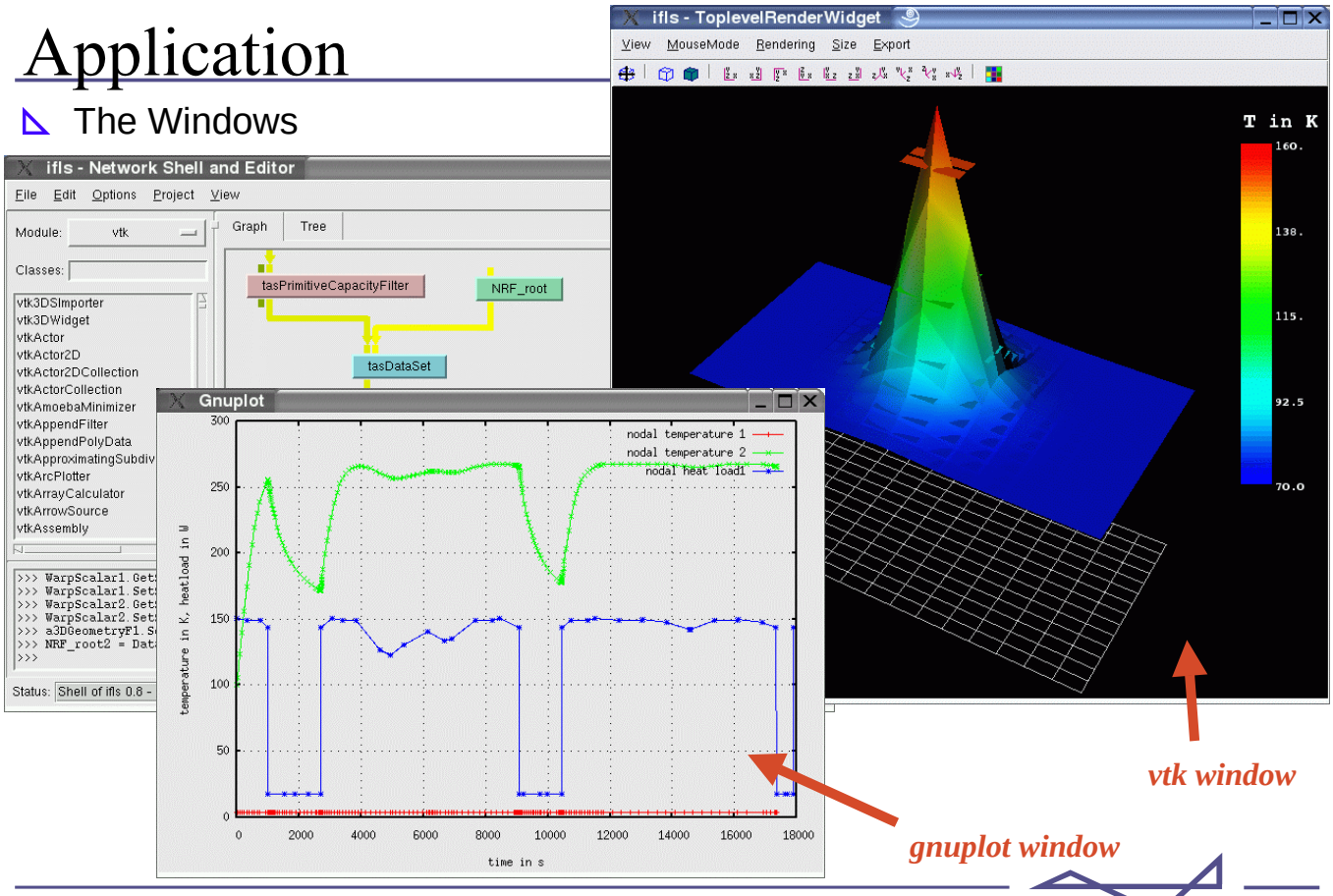
- **nrfSimpleDatacube** to **numpy** conversion
- Further **numpy**-processing
- **numpy** plots with **gnuplot (pipe)**



IFL

Application

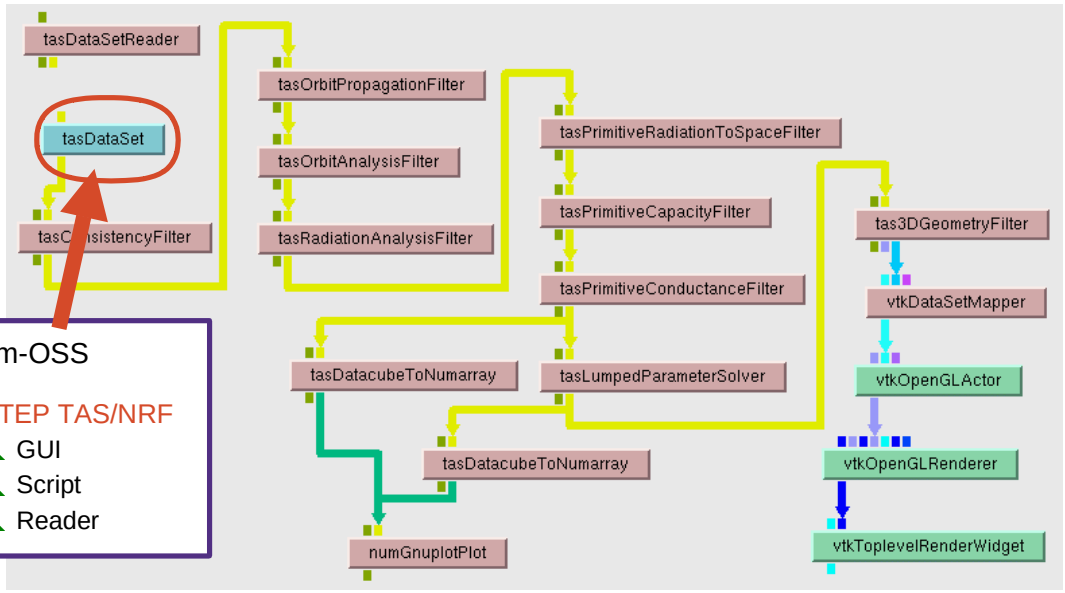
- The Windows



IFL

Application

- DataSet Generation

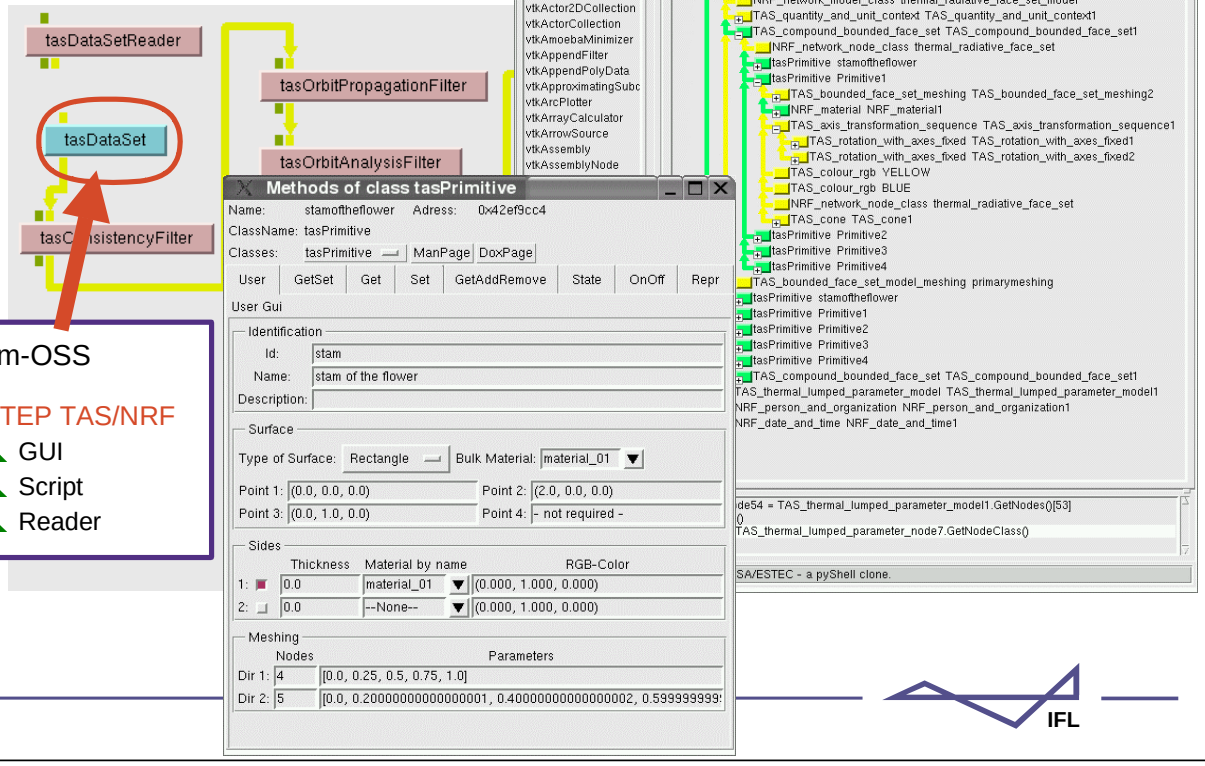


- Therm-OSS
  - STEP TAS/NRF
    - GUI
    - Script
    - Reader

IFL

Application

- DataSet Generation

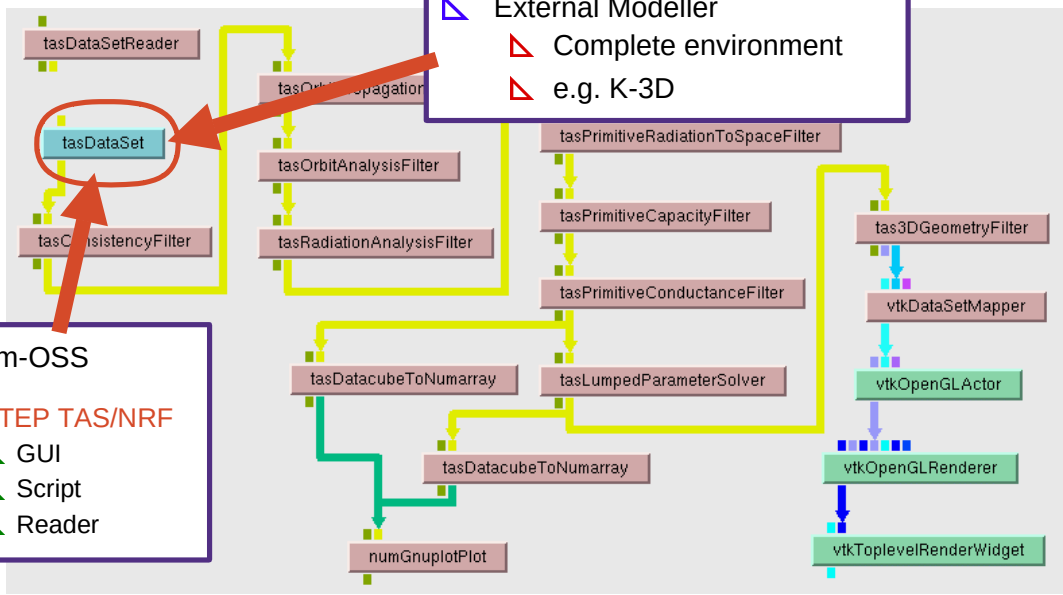


- Therm-OSS
  - STEP TAS/NRF
    - GUI
    - Script
    - Reader

IFL

Application

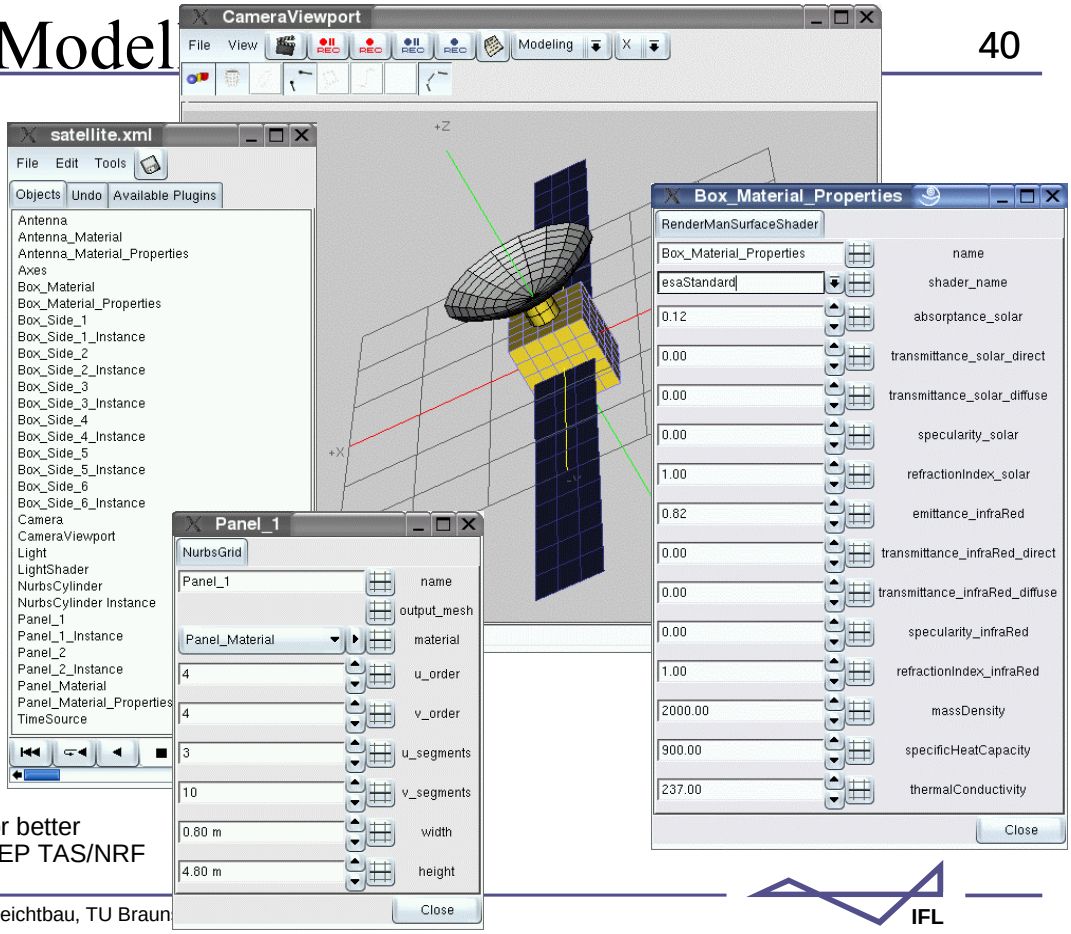
- DataSet Generation



- External Modeller
  - Complete environment
  - e.g. K-3D
- Therm-OSS
  - STEP TAS/NRF
    - GUI
    - Script
    - Reader

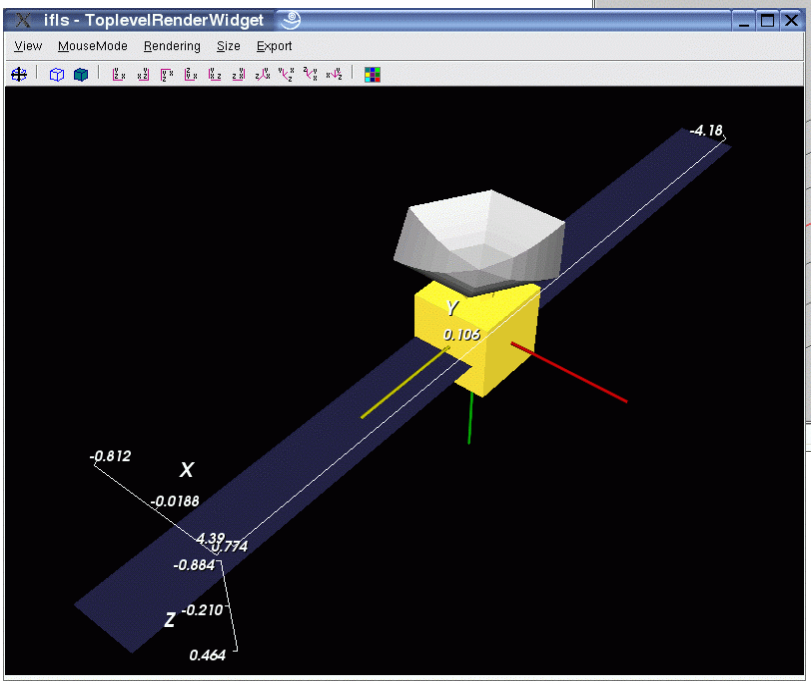
External Model

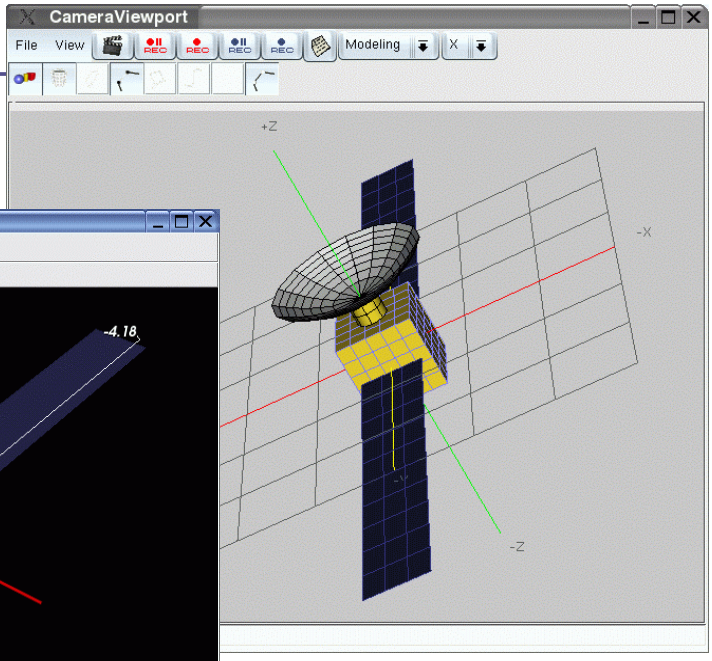
- K-3D
  - Primitives
    - RenderMan
    - NURBS
    - Meshes
  - Hierarchy
    - Graph
  - Material
    - Shader
  - Therm-OSS
    - **Reader**
    - **Writer**
    - ***Filter***
    - ... ***Plug-Ins*** for better support of STEP TAS/NRF



External Modeller

- K-3D
  - STEP-TAS:





Conclusions
Outlook

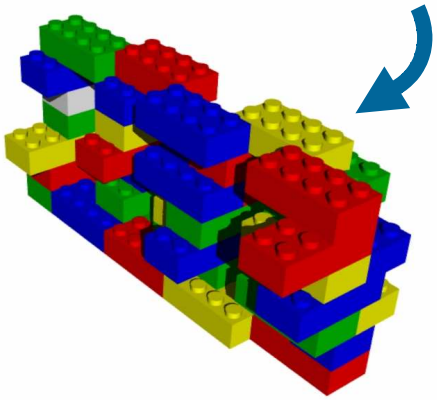
Lessons learned

- *Powerful tools for design, analysis and tool integration available as OSS*
- *Finding and evaluation of OSS is not easy*
- *License problem (free for research, not for commercial)*
- *Tools are 80-90 % satisfying*
- *Rapidly changing versions*
- *Don't think straight*

- *Open Source approach of a tool integration platform is successful*
- *Everything runs on Linux and Windows*
- *Installation from source*

www.Therm-OSS.org

Component based architecture



Therm-OSS development

- More comfortable for the simple user
- More sophisticated components
- *New components: Predict and RenderPark*

Objectives of the Therm-OSS Project

- To assess how OSS can be used to build applications
  - Architecture, components and processes of this prototype
- To provide to developers a useful source of reference for their developments
  - OSS-Survey, experiences and ideas
- To assess whether the OSS approach could be useful as a distributed model for end-users
  - Still open … interest for end-users ?