Robust industrial model exchange between ESARAD and THERMICA with STEP-TAS

Hans Peter de Koning, Simon Appel, David Alsina Orra
(ESA/ESTEC D/TOS-MCV)

with some contributions from Simulog and Graitec

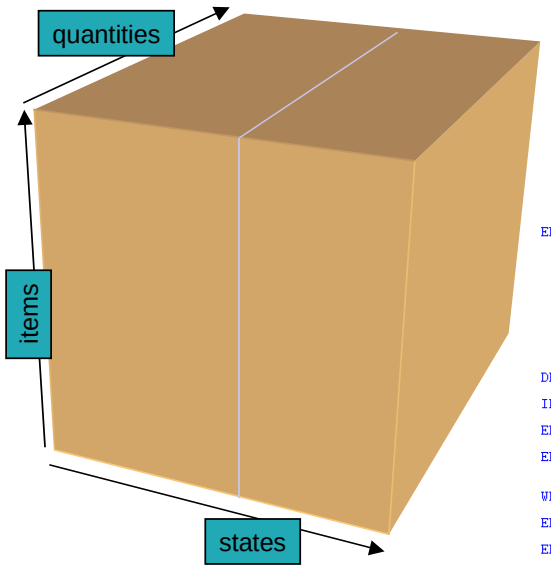
Topics

- Brief recap of STEP-NRF and STEP-TAS protocols
- Why TASverter ?
- Scope and purpose of TASverter
- Development approach
- Updates to NRF and TAS protocols
- Current status
- Outlook

Recap STEP-NRF (1)
(Network-model Results Format)

- Generic, discipline-independent exchange of models & results
  - Model definition, using a discrete network representation
    - Supports model/submodel hierarchy
  - Results data, produced in analysis, test or operation
  - Meta-data, which records details of actual analysis, test or operation performed
- Only discrete observations
  - I.e. sampled results at discrete locations for discrete states of the model / object under observation, no support for continuous fields or similar
- Any property value has explicit (physical) quantity and unit
- Data model designed to cope efficiently with large amounts of results
  - Built-in support for scalar, vector, matrix, tensor data

Recap STEP-NRF (2)
(Network-model Results Format)



Central NRF data structure is the 'datacube'

- each element of the cube is a scalar, vector or tensor property for a specific (item, quantity, state)
- state quantity is normally time or frequence
- 'simple' and 'advanced' SUBTYPEs, respectively for literal and generalised functional property values

```
ENTITY NRF_any_datacube
  ABSTRACT SUPERTYPE OF( ONEOF( NRF_simple_datacube, NRF_advanced_datacube ) );
  -- a (hyper)cubic value space in 3 dimensions: state, quantities and items
  -- the ordering of the nested list of values is determined by value_order
  id                 : NRF_identifier;
  security_class     : OPTIONAL NRF_security_classification_level;
  value_order        : NRF_datacube_order_type;
  quantity_base      : NRF_quantity_list;
  item_base          : NRF_observable_item_list;
  state_quantity     : NRF_scalar_quantity;
  state_base         : NRF_state_value_list;
DERIVE
  number_of_states : INTEGER := SIZEOF( state_base.state_values );
INVERSE
  dataset            : NRF_dataset FOR datacubes;
END_ENTITY;

ENTITY NRF_simple_datacube
  SUBTYPE OF( NRF_any_datacube );
  values             : LIST [1:?] OF NRF_real_or_integer_literal;
WHERE
  wr1: SIZEOF( values ) = quantity_base.number_of_elements * SIZEOF(item_base.items) * number_of_states;
END_ENTITY;

ENTITY NRF_advanced_datacube
  SUBTYPE OF( NRF_any_datacube );
  values             : LIST [1:?] OF NRF_any_real_or_integer_value;
WHERE
  wr1: SIZEOF( values ) = quantity_base.number_of_elements * SIZEOF(item_base.items) * number_of_states;
END_ENTITY;
```

Recap STEP-TAS (Thermal Analysis for Space)

- STEP-based application protocol
  - Initial scope: Exchange of thermal-radiative models for space, including rigid body kinematics and orbit / attitude / orientation specification
    - Geometry represented by bounded face model with minimal topology (compatible with AP 203 CC4)
  - Extended scope: Exchange of thermal lumped parameter network models
    - Targetting exchange between various SINDAs, ESATAN
  - STEP-TAS is a pure superset of STEP-NRF (Used as 'integrated resource')
  - Developed since 1996 mainly on ESA funding, supported by CNES, NASA
    - Originally full ARM / AIM according to ISO TC184/SC4 procedures
    - Accompagnying STEP-TAS library (C and F77 API) provided since 1998
    - Many tools implemented prototype or industrial beta converters

Why TASverter ?

- STEP-TAS converters did not deliver industrial solution up to now
  - Only ESARAD, THERMICA and Thermal Desktop have STEP-TAS exchange included in industrial releases
  - Exchange is slow, often not reliable, and fails for large models
- Existing STEP-TAS architecture had too many layers
  - Bad performance (CPU / elapsed time)
  - Inefficient memory usage, huge memory requirements
  - Expensive to verify and maintain
  - Difficult to distribute on multiple platform/compiler combinations
- However principle of providing protocol + library was very good and should be retained!

'STEP-TAS high level API' (C and F77)
STEP-TAS ARM
STEP-TAS AIM
SDAI-C (COTS, late binding)
Vendor specific repository handler (COTS)

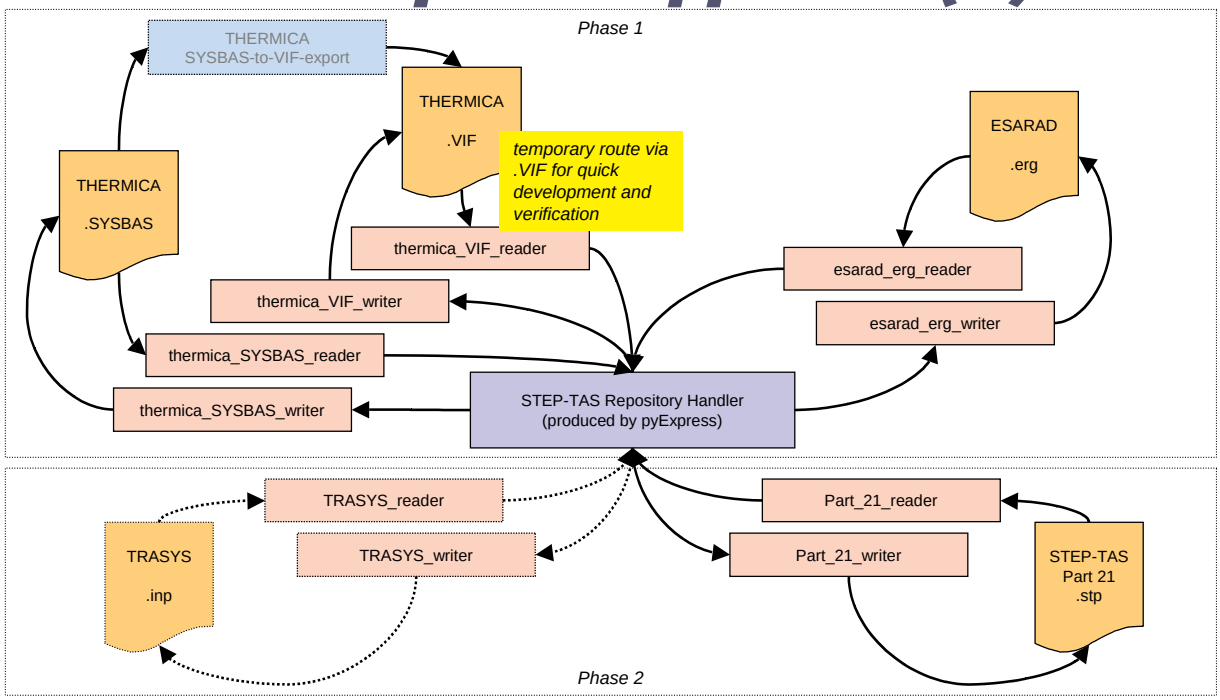
Scope and purpose of TASverter

- TASverter is an initiative of ESA/ESTEC D/TOS-MCV to:
  - Offer users finally a properly working solution for exchange of thermal models
    - First between major European analysis tools ESARAD and THERMICA
  - Remove complicated dependency on (at least) 4 developers
    - STEP-TAS and STEP library developers, Tool X and Tool Y developers
  - Produce a fully functional (open source) framework for validation and verification of STEP-based data exchange protocols and implementations
  - Lay a solid basis for the future
    - Low threshold for implementation
    - Maintainable and cost-effective
    - Ensure long term availability (no dependence on closed 3rd party software)

Development approach (1)

- Implementation in pure Python (v2.2)
  - Following positive experience with earlier 'ad-hoc' converter developments
- Internal data storage uses STEP-TAS (ARM) data model
  - Implemented in 'STEP-TAS Repository Handler', which is largely generated automatically with pyExpress from the STEP-TAS EXPRESS schema
- For each supported tool/format a 'reader' and a 'writer' is created
- Full testsuite built up alongside development
  - Unit, integration and large model testcases under configuration control
- Integrated validation and fine-tuning of STEP-NRF and STEP-TAS
  - Goal is recreation of models which are understandable and editable by humans
  - Efficient update cycle is possible with pyExpress STEP-TAS library generator

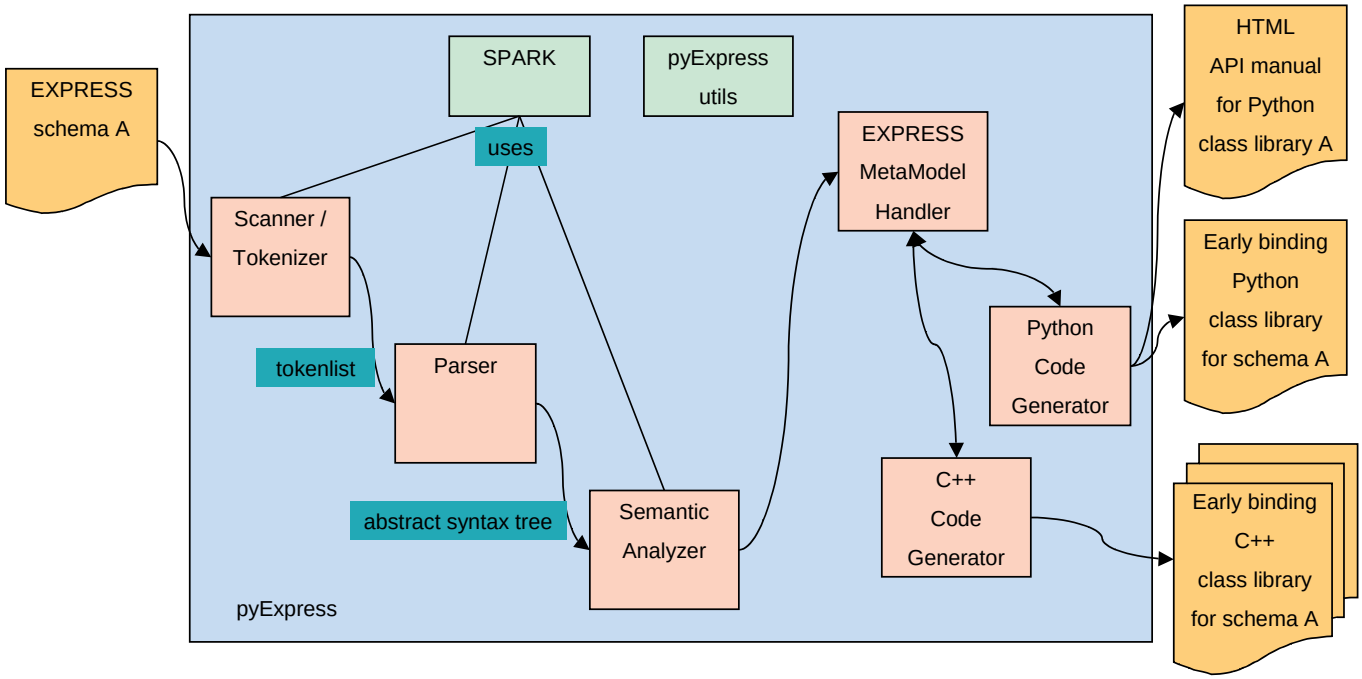
Development approach (2)



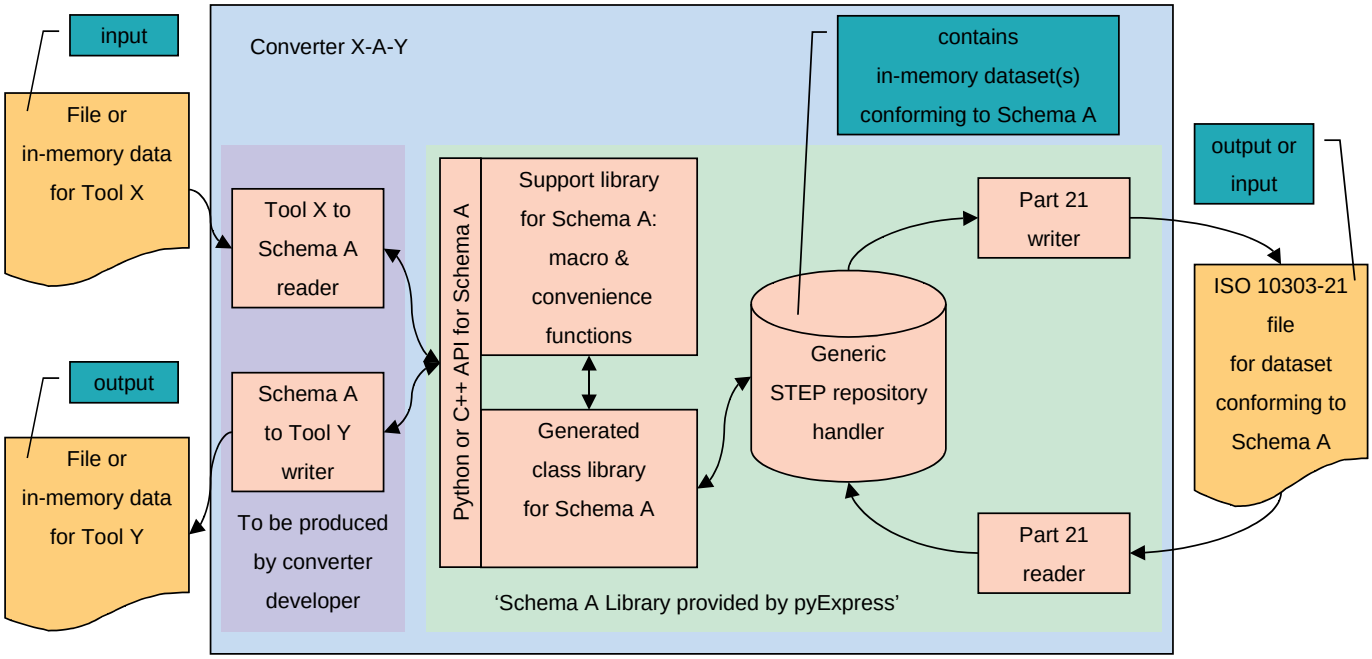
Development of pyExpress

- Provide a STEP converter development environment, that
  - can be used by converter developer with minimal STEP knowledge
  - can be used as a Rapid Application Development tool for prototyping and near-real time validation and refinement of application protocols
  - has very strong string manipulation capabilities
  - maps well onto EXPRESS object oriented data models
  - leads to industrially robust converters, with acceptable performance and memory requirements, also for large models
- pyExpress is a STEP/EXPRESS compiler / code generator
  - Generates Python class library for implementation of converter in Python
  - Generates C++ class library for implementation of converter in C++

pyExpress architecture



Outline of converter based on pyExpress

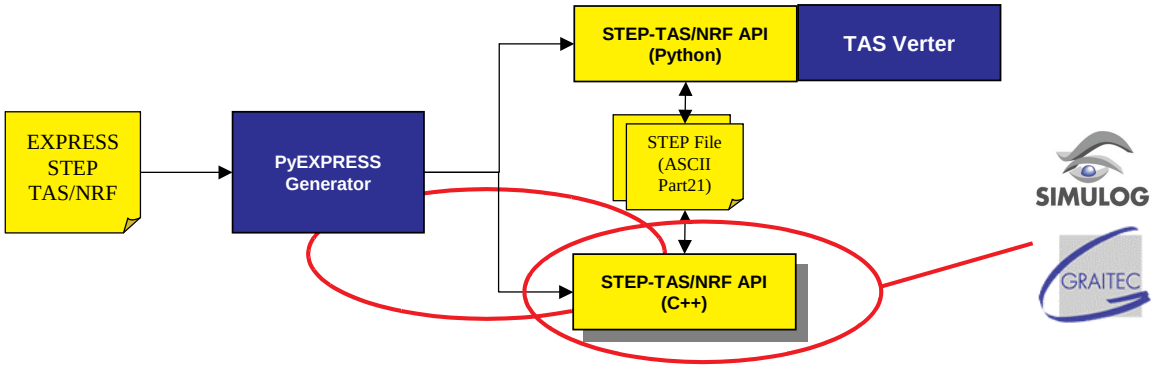


EXPRESS → Python Example (1)

```
ENTITY TAS_kepler_parameter_set;
  semi_major_axis                    : NRF_length_measure;
  eccentricity                       : REAL;
  inclination                        : NRF_plane_angle_measure;
  right_ascension_of_ascending_node  : NRF_plane_angle_measure;
  argument_of_periapsis              : NRF_plane_angle_measure;
  true_anomaly_at_start              : NRF_plane_angle_measure;
WHERE
  wr1: semi_major_axis >= 0.0;
  wr2: eccentricity >= 0.0;
  wr3: { -180.0 < inclination <= 180.0 };
  wr4: { -360.0 < right_ascension_of_ascending_node <= 360.0 };
  wr5: { 0.0 <= argument_of_periapsis < 360.0 };
  wr6: { -360.0 < true_anomaly_at_start <= 360.0 };
END_ENTITY;
```

```
class TAS_kepler_parameter_set(ExpressEntity):
    _attr_list=[
        ['semi_major_axis', 'NRF_length_measure'],
        ['eccentricity', 'REAL'],
        ['inclination', 'NRF_plane_angle_measure'],
        ['right_ascension_of_ascending_node', 'NRF_plane_angle_measure'],
        ['argument_of_periapsis', 'NRF_plane_angle_measure'],
        ['true_anomaly_at_start', 'NRF_plane_angle_measure']]
    _attr_id_maxlen=33

    def __init__(self,
        semi_major_axis                    = None,  # NRF_length_measure
        eccentricity                       = None,  # REAL
        inclination                        = None,  # NRF_plane_angle_measure
        right_ascension_of_ascending_node  = None,  # NRF_plane_angle_measure
        argument_of_periapsis              = None,  # NRF_plane_angle_measure
        true_anomaly_at_start              = None): # NRF_plane_angle_measure

        ExpressEntity.__init__(self)

        assert _istype_NRF_length_measure(semi_major_axis)
        assert _istype_REAL(eccentricity)
        assert _istype_NRF_plane_angle_measure(inclination)
        assert _istype_NRF_plane_angle_measure(right_ascension_of_ascending_node)
        assert _istype_NRF_plane_angle_measure(argument_of_periapsis)
        assert _istype_NRF_plane_angle_measure(true_anomaly_at_start)

        self.semi_major_axis = semi_major_axis
        self.eccentricity = eccentricity
        self.inclination = inclination
        self.right_ascension_of_ascending_node = right_ascension_of_ascending_node
        self.argument_of_periapsis = argument_of_periapsis
        self.true_anomaly_at_start = true_anomaly_at_start

    def set_semi_major_axis(self, semi_major_axis):
        assert _istype_NRF_length_measure(semi_major_axis)
        self.semi_major_axis = semi_major_axis
        self.compute_derived_attributes()

    def get_semi_major_axis(self):
        return self.semi_major_axis
```


PyEXPRESS automatic generation



- Complete Open Source code of new STEP-TAS/NRF C++ API, including STEP-21 write/read
  - automatically generated from EXPRESS schema
- Validated on :
  - Windows NT4/2000
  - Silicon Graphics /Irix 6
  - SUN/Solaris 8

New STEP-TAS/NRF C++ API Example

```
ENTITY TAS_triangle
SUBTYPE OF
(primitive_bounded_surface);
p1 : geometric_construction_point;
p2 : geometric_construction_point;
p3 : geometric_construction_point;
END_ENTITY;
```

```
class  Tas_triangle : public  Tas_primitive_bounded_surface
{
public:
//construction destruction
Tas_triangle();
~Tas_triangle();
//attributes
Tas_geometric_construction_point* get_p1() const;
void set_p1(const Tas_geometric_construction_point *& p1)
throw (StepException);
Tas_geometric_construction_point * get_p2() const;
void set_p2(const Tas_geometric_construction_point *& p2)
throw (StepException);
Tas_geometric_construction_point * get_p3() const;
void set_p3(const Tas_geometric_construction_point *& p3)
throw (StepException);
…
```


Updates to NRF and TAS protocols (1)

- Removed AIM and mapping table, focussed on robust ARM
  - Interoperability with AP203 was not achieved in practice
  - Better do an executable AP203/TAS converter (retained mapping table for this)
  - Cost of full AIM implementation, verification and maintenance too high
- Major clean-up and replacement of unclear terminology
  - Resolved many issues collected over the years
  - Includes artificial constructs in ARM from original AIM/GR mapping
- Changed TAS navigational structure from bottom-up to top-down
  - Much easier / more natural to use in OO repository API
- Revalidated relationships for TAS geometry, meshing, thermal-radiative faces and made data model more consistent

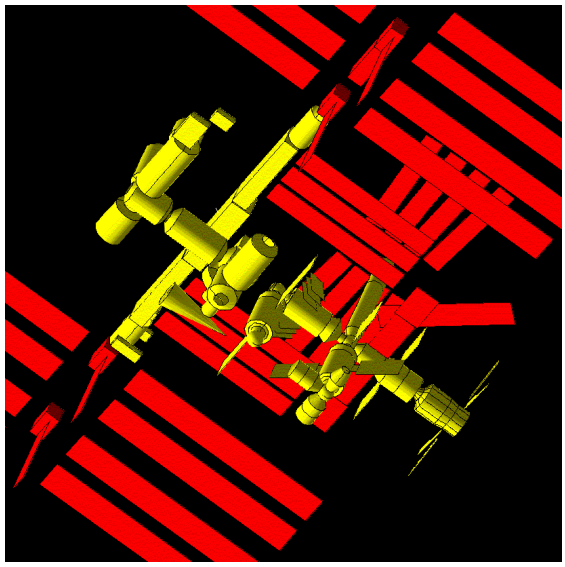
Updates to NRF and TAS protocols (2)

- Updated NRF definitions for ‘datacube’, quantities and properties
  - Permitted all permutations of ordering (item, quantity, state)
  - Added a dedicated datacube for material properties
- Started to move protocol documents from MS Word to XHTML
- Revalidated all WHERE and RULE constraints
- Made all INVERSE attribute definitions consistent
- Consequence is that new STEP-TAS (ARM) Part 21 files are not compatible with previous STEP-TAS (AIM) Part 21 files
  - Not a serious problem since STEP-TAS was not yet really in industrial use
  - Last chance for this kind of updates

Current TASverter status

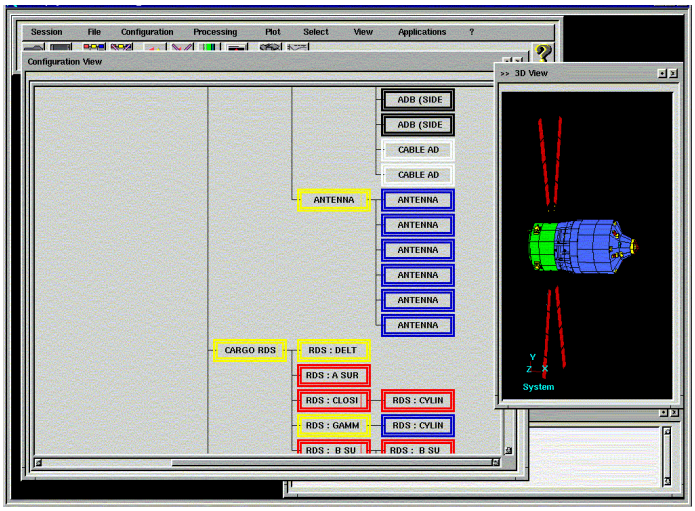
- Started in October 2002; 4th release made per 10 Oct 2003
  - Self-contained Windows, Solaris, Linux, Irix executables
    - No need to install Python
  - Free download from http://www.estec.esa.int/thermal/tools/tasverter.html
  - THERMICA .VIF and .SYSBAS readers/writers
  - ESARAD .erg reader/writer
  - STEP-TAS Part 21 reader/writer
  - Configuration controlled testsuite with unit and large model testcases, including fully automated run scripts for verification and regression testing
- CIGAL-2 reader/writer in progress
  - By Alcatel Space + OpenCascade (with some assistance from ESTEC)

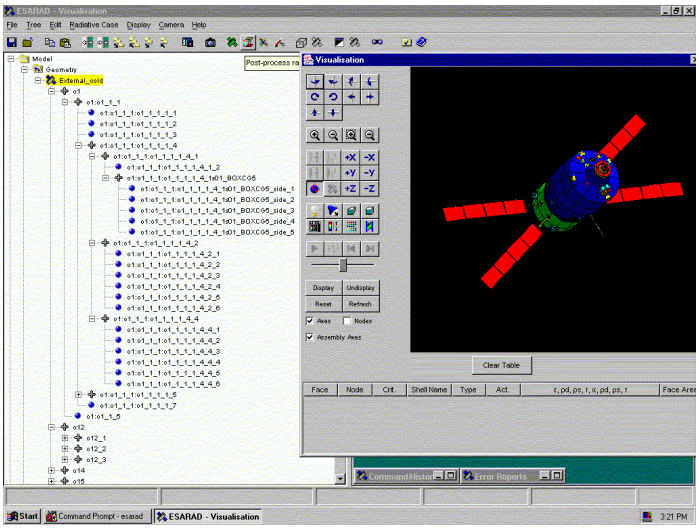
TASverter example 1: ISS_cold
THERMICA to ESARAD



711 thermal-radiative surfaces, converted in less than 15 seconds.

TASverter example 2: ATV
THERMICA to ESARAD





1700 thermal-radiative surfaces, converted in less than 25 seconds.

Model hierarchy and coordinate transformations fully retained.

TASverter example 2: ATV THERMICA to ESARAD

```
<1,1,1,1,4,4,2>    ANTENNA LAT MAIN BODY
$INFO      COLOUR=BLUE
$C_PROPERTY
     C_COATING=PSG_120_FD_cold
$TSHAPE    CYLINDER P1=(       .0,       .0,    -4.380) &
                    P2=(       .0,       .0,    -4.400) &
                    P3=(    0.155,       .0,    -4.380) &
                    DIAM=0.154
           MESH     NODE=(1,1) ELEM=(1,1) SIDE=POS
$THERM     NAME=(9966)
$AXIS      TRAX=0.04  TRAY=-0.333


<1,1,1,1,4,4,3>    ANTENNA UPP MAIN BODY
$INFO      COLOUR=BLUE
$C_PROPERTY
     C_COATING=PSG_120_FD_cold
$TSHAPE    DISC     P1=(       .0,       .0,    -4.400) &
                    P2=(       .0,       .0,    -4.480) &
                    P3=(    0.155,       .0,    -4.400) &
                    DIAM1=0.154  DIAM2=0.045
           MESH     NODE=(1,1) ELEM=(1,1) SIDE=POS
$THERM     NAME=(9965)
$AXIS      TRAX=0.04  TRAY=-0.333
```

Original THERMICA .SYSBAS

```
SHELL o1_1_1_1_4_4_2s01;
o1_1_1_1_4_4_2s01 = SHELL_SCS_CYLINDER(
    label   = "ANTENNA LAT MAIN BODY <1,1,1,1,4,4,2>",
    radius  =  7.70000000000000e-002,
    hmax    =  2.00000000000005e-002,
    hmin    =  0.00000000000000e+000,
    angmax  =  3.60000000000000e+002,
    angmin  =  0.00000000000000e+000,
    side1   = "ACTIVE",
    side2   = "INACTIVE",
    opt1    = PSG_120_FD_cold,
    nbase1  = 9966,
    ndelta1 = 1,
    colour1 = "BLUE",
    colour2 = "BLUE",
    nodes1  = 1,
    nodes2  = 1,
    ratio1  = 1.000000,
    ratio2  = 1.000000,
    thick   = 0.0);

o1_1_1_1_4_4_2s01 =
    ROTATE (object_name = o1_1_1_1_4_4_2s01,
            x_ang =  1.80000000000000e+002,
            y_ang =  0.00000000000000e+000,
            z_ang =  0.00000000000000e+000);

o1_1_1_1_4_4_2s01 =
    TRANSLATE (object_name = o1_1_1_1_4_4_2s01,
               x_dist =  0.00000000000000e+000,
               y_dist =  0.00000000000000e+000,
               z_dist = -4.38000000000000e+000);

SHELL o1_1_1_1_4_4_2;
o1_1_1_1_4_4_2 =
          (o1_1_1_1_4_4_2s01);

o1_1_1_1_4_4_2 =
    TRANSLATE (object_name = o1_1_1_1_4_4_2,
               x_dist =  4.00000000000000e-002,
               y_dist =  0.00000000000000e+000
```

Generated ESARAD .erg

Verification Test Suite

- More than 200 unit tests
  - Documented as a website
    - with naming convention for subdirectories per testcase
    - actual and reference results for regression testing
  - Fully scripted to run and be diff-ed automatically
- Real model tests
  - ATV model
  - METOP C/D full spacecraft model
  - ISS model
  - Herschel and Planck full spacecraft models

Short term priorities

- STEP-TAS Verification Tool (in progress)
  - Semi-automatic verification of STEP-TAS Part 21 files
  - 3D visualisation for visual inspection
  - Extraction of key characteristics of exchanged model
    - Total surface area, surface area per aggregation level
    - Overall geometric envelop
    - Materials and material properties
    - Number of faces, surfaces
    - Number of thermal lumped parameter nodes, node ranges
  - Goal is to enable ESA to verify (certify) correctness of STEP-TAS datasets produced by different converters and to isolate cause of possible errors
- Up-to-date STEP-NRF and STEP-TAS documentation (in progress)
  - Including converter implementation examples

Outlook (short term)

- Submit STEP-NRF and STEP-TAS protocols to ISO TC 184 / SC 4 for ballot as ISO PAS or TS
- Publish schemas and tools/toolkits in open source (harmonisation)
  - pyExpress and TASverter
  - develop full capability pyExpress with University of Manchester
  - STEP-TAS and STEP-NRF schemas, Python and C++ libraries
- Provide and verify pyExpress generated C++ libraries for STEP-TAS
  - If requested by thermal analysis tool vendors
- ESA development STEP-SPE (Space Environmental Analysis)
  - Contract awarded, real work starting 27-Oct-2003
  - Extends TAS for micro-meteorites/debris, contamination, atomox, radiation …

Outlook (longer term)

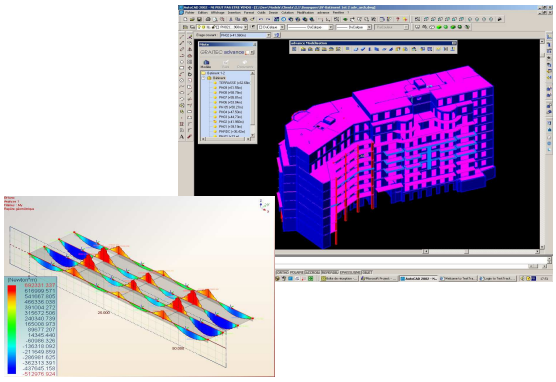
- Promote implementation STEP-TAS in US, Canadian tools
  - TMG, Thermal Desktop, TSS, ...
- Possibly extend TASverter with new reader/writer plug-ins
  - Transform existing TRASYS/ESARAD converter to TRASYS reader/writer
  - Transform existing SINDA85/ESATAN converter to SINDA85 reader and ESATAN writer
  - Add more SINDA/ESATAN-like readers / writers
  - Add AP203 reader/writer, with primitive shape recognition capability
    - Can be derived from existing AP203/ESARAD converter plus old TAS AIM mapping; possibly add facetting of remaining NURBS surfaces
  - Construct HDF5 mapping and libraries for STEP-NRF

Related: STEP in the Building Industry

- Characteristic of Building Industry
  - A lot of independent large constructors and SME involved in one (big) building
  - Building becoming more and more complex
    - Structural, thermal, acoustic, electricity…
    - Hard regulation
  - CAD : Very large objects models

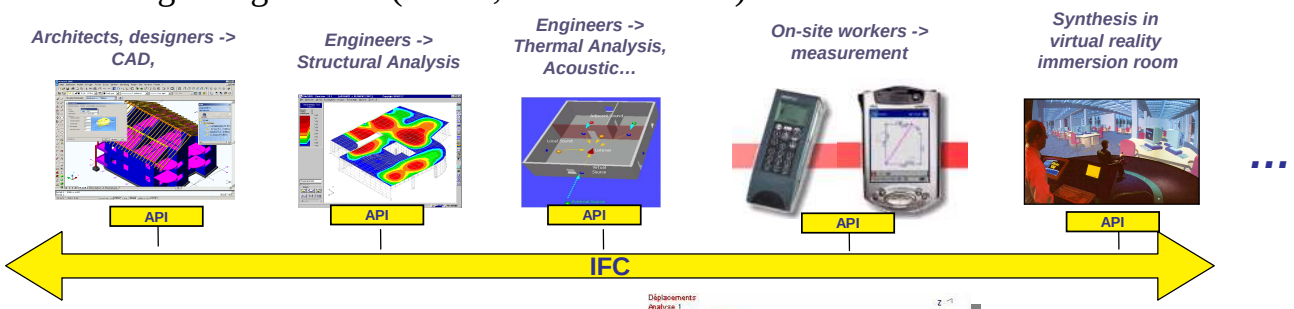
- STEP for building = IFC (Industry Foundation Classes)
  - Based on EXPRESS and STEP-21
  - Building dedicated integrated model :
    - architecture, materials, structural, thermal, HVAC…

- SIMULOG partnership, for STEP, CAD, Post-pro with GRAITEC
  - 3rd European software editor in the building industry
    - **+3500** customers
    - **+8000** licences CAD/analysis tools
    - **+100 000** buildings designed

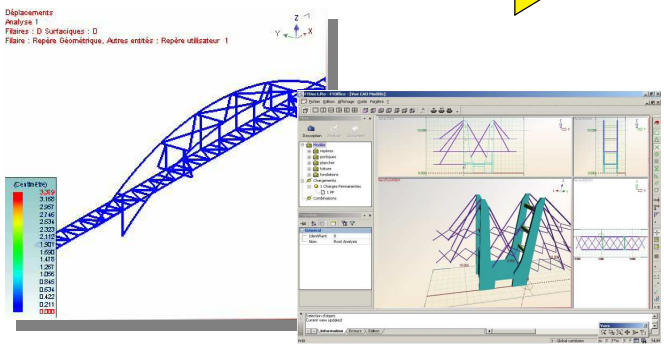


PyExpress IFC-API (C++) used in 2 major projects

- Building Design Chain (CSTB, Mediaconstruct)



- IFC-BRIDGE (SETRA)



Acknowledgements

- ESTEC team: Hans Peter de Koning, Simon Appel, David Alsina
- Contractor Simulog (France): Olivier Pailles, Arnaud Klinger
- Subcontractor GRAITEC (France): Eric Lebègue and co-workers
- ESARAD / ALSTOM Power (UK): Julian Thomas, David Scurrah
- THERMICA / Astrium SAS (France): Marc Jacquiau