STEP-TAS & TASverter from the software developer's point of view

Hans Peter de Koning
(ESA/ESTEC D/TEC-MCV)

Topics

- Why open data exchange standards?
- Overview of general data exchange standardisation for space industry
- Short history of STEP-TAS development
- Main elements of the STEP-TAS standard and implementation software
- Supporting implementation software: pyExpress and TASverter
- Further development and formal standardisation schedule

Why open data exchange standards? (1)

Reliable and easy-to-use product data exchange is essential

in order to achieve efficient and cost-effective industrial product development processes

- Prescription of single CAx tools (per discipline) in space projects is not effective
  - Project teams involve many partners and are often multi-national
  - Each organization should have the possibility to optimize its own processes
  - Support for multiple tools within one organization is costly – licenses, training
  - Competition between tool developers is healthy, yields better tools, promotes innovation

Why open data exchange standards? (2)

- Direct conversion between tools may provide a short term solution
  - But not sustainable over longer term: maintenance cost and reliability problems
  - Converter developer controls and masters only one side of interface
  - N tools require N*(N-1) converters for complete exchange capability
  - Large duplication of effort
- Data exchange via open standards is the rational long-term solution
  - Stability of open standard can be guaranteed by independent international body
  - Both sides of interface are fully visible to converter developer
  - N tools require 2N converters for complete exchange capability
  - However places very severe requirements on the quality and completeness of the standard and its supporting implementation software
  - Drawback is that open standard has to address lowest common denominator, therefore loss of information after transfer can not always be prevented

Requirements on open data exchange standards and implementation technology

- Shall be reliable
- Shall be easy to use and understand by end-users
  - Absolute minimum number of transfer parameter settings
- Shall be rigorously verifiable
- Shall be complete and self-contained – yet as simple as possible
- Shall be designed for extension with full backwards compatibility
- Shall be portable – no computer platform dependencies
- Shall avoid dependence on third party proprietary software
- Shall be designed for low cost implementation and maintenance
  - Shall minimize required investments from tool/converter developers

Additional uses of open data exchange standards

- Long term archiving of models and results
- Well-controlled migration path from existing tools to next generation tools
  - Enlarges possibilities for end-users – stimulates competition between developers
  - Major benefits for rigorous verification of new software tool
- Tool-independent definitions of benchmark problems
- Developments sponsored from public funding (e.g. ESA) could be done against open standard's programming interface
  - Enables sharing of R&D results between different tool developers
- Custom utilities could be created efficiently using the open standard's programming interface

Open data exchange standards relevant for aerospace (1/2)

ISO 10303 (STEP = *Standard for the Exchange of Product model data*)

- ✓ Part 11: EXPRESS data modelling language
- ✓ Part 21: Physical file
- ✓ Parts 22,23,24,26: C, C++, Java Programming I/Fs
- ✓ Part 28: Link with XML and XMI data transfer ⇐
- ✓ Parts 4x: Generic Resources: product structure, geometry, topology, …
- ✓ Parts 5x: Basic blocks engineering analysis: mathematical spaces, functions, structured/unstructured meshings, properties, results data, …
- ✓ PDM schema
- ✓ AP203: PDM + 3D explicit shapes ⇐
- ✓ AP209: FE analysis
- ✓ AP210: Electronic assemblies (avionics, PCBs, …)
- ✓ AP212: Electro-technical (harnesses, …)
- ✓ AP214: Automotive (AP203 + CSG, kinematics, …)
- ✓ AP221: Process plant (registries, STEPLib, multi-language)
- ✓ AP232: Technical data packages
- ✓ AP233: Systems engineering ⇐
- ✓ AP237: CFD data

Basic ISO standards

- ✓ ISO 31 & 1000: Quantities and units, SI
- ✓ ISO 8879 SGML: Standard Generalized Markup Language

ISO 13584 / PLIB

- ✓ STEP-compatible parts libraries ⇐

STEP-based standards (*developed by ESA*)

- ✓ STEP-TAS: Thermal Analysis for Space
- ✓ STEP-NRF: Network-model Results Format ⇐

Open data exchange standards relevant for aerospace (1/2)

W3C standards

- ✓ HTTP and URI/URL
- ✓ HTML & XHTML
- ✓ XML: eXtensible Markup Language ⇐
- ✓ MathML: Mathematical Markup Language ⇐
- ✓ XSL & XSLT & XPath: eXtensible Stylesheet Language and XSL Transformations & Path
- ✓ XML-Schema ⇐
- ✓ RDF: Resource Description Framework ⇐ "Semantic Web"
- ✓ OWL: Web Ontology Language ⇐
- ✓ DOM: Domain Object Model
- ✓ SOAP: Simple Object Access Protocol ⇐ "Web Services" based on HTTP and XML
- ✓ PNG: Portable Network Graphics
- ✓ SVG: Scalable Vector Graphics

IETF

- ✓ LDAP: Lightweight Directory Access Protocol ⇐

OMG standards

- ✓ UML: Information System Modelling ⇐
- ✓ OCL: Object Constraint Language ⇐
- ✓ SysML: System Engineering ⇐
- ✓ CORBA: OO Distributed Processing
- ✓ MDA: Model Driven Architecture ⇐
- ✓ XMI: XML Metadata Interchange ⇐
- ✓ MDTF: Manufacturing Domain Task Force

Web 3D Consortium

- ✓ VRML: Virtual Reality Modeling Language ISO/IEC 14772

Public domain standards

- ✓ HDF5: Hierarchical Data Format (NCSA) ⇐
- ✓ NetCDF: Network Common Data Form (UCAR)

ECSS E-10 Part 7 "Product data exchange"

- ECSS E-10 Part 7 "Product data exchange" in final stage of publication
  - Available October 2004 from http://www.ecss.nl
  - 'Umbrella' standard providing a central point of reference for all product data exchange standards applicable to space projects – a tailorable list of references to standards to exchange data from a discipline A to a discipline B
  - Will be updated on a regular basis (probably once a year)
- ECSS E-10 Part 9 Engineering Database (in progress)
- ECSS E-10 Part 13 Modelling & Simulation (in progress)

Example clause ECSS E-10 Part 7

The specific requirements are given as subclauses under subclause 4.3, and written in a structured way. For each discipline listed in Table 1 there is a level 3 subclause (4.3.x) which designates the source discipline. Each source discipline subclause contains a collection of applicable destination disciplines, referenced as 4.3.x.y. The body of all specific requirement subclauses adheres to the following template:

Table (example)

Reference	Source representation	Destination representation	Product data to be transferred	Standard reference to apply
4.3.x.y				

Layers in different standard families

Standard family	ISO 10303 (STEP)	W3C XML	W3C Semantic Web	W3C Ontology	OMG UML/MDA
Origin	Mechanical engineering	Structured web data	Structured web data with meaning	Structured web data capturing knowledge	Software engineering
Data structure definition	ISO 10303-11 EXPRESS	DTD XML Schema	RDF Schema (uses XML Schema datatypes)	OWL (Lite/DL/Full) (builds on top of RDF Schema)	UML OCL XMI
File exchange	ISO 10303-21 clear text encoding ("STEP file") ISO 10303-28 XML encoding ISO 10303 Binary (in progress, possibly HDF5)	XML Unicode encoding (e.g. UTF8) XML/Binary (in progress)	RDF-XML	OWL-XML	-
Data access API	ISO 10303-22 SDAI ISO 10303-23 C++ ISO 10303-24 C ISO 10303-27 Java	DOM SAX	RDF library (various open source)	OWL library (various open source e.g. Jena)	Generated from UML model

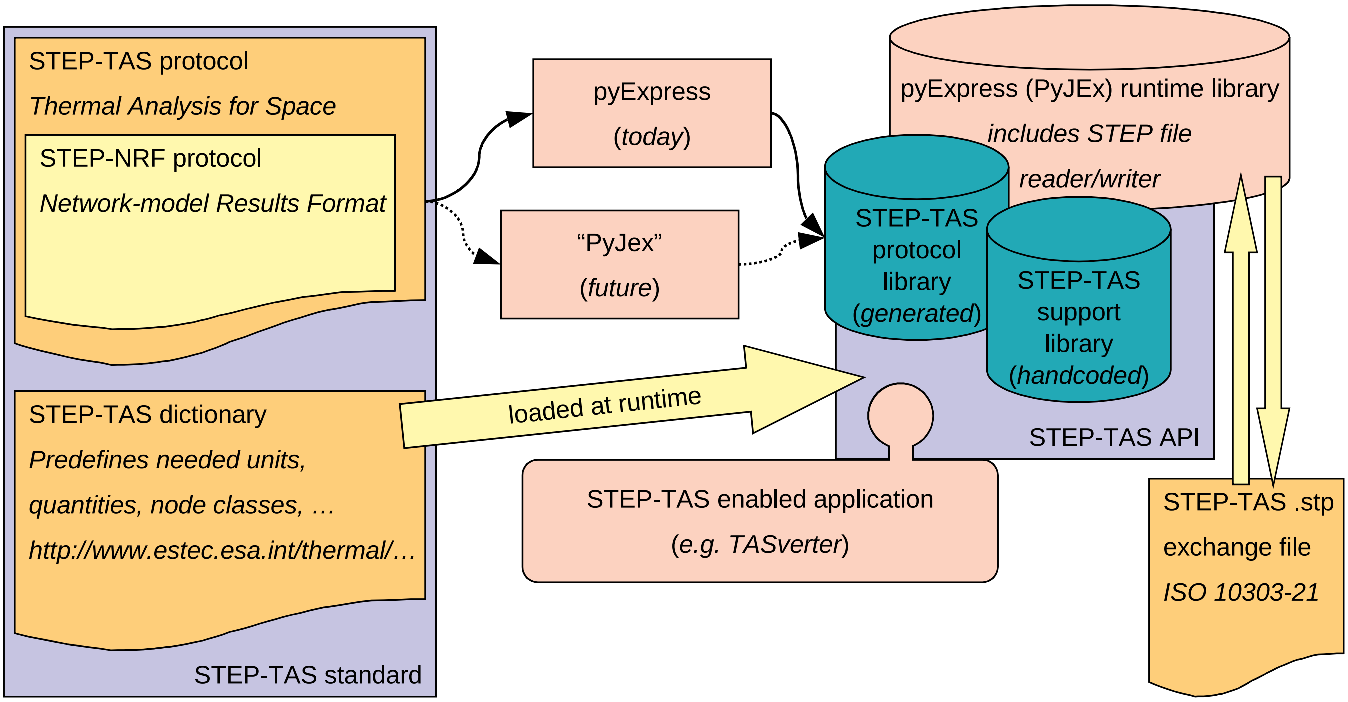
Short history of STEP-TAS development (1)

- In 1995 ESA selected ISO 10303 (STEP) as the basis for the *Thermal Analysis for Space* data exchange standard
  - Nowadays one would possibly select an XML based approach, but in 1995 XML was not yet around and even now XML and XML/Schema still lack some of the more advanced features of the STEP architecture – in addition STEP and XML are being consolidated: ISO 10303-28 (released in 2002) defines how to map STEP to XML and back
- The STEP-TAS standard consists of 3 parts:
  - The NRF (Network-model Results Format) protocol (with EXPRESS schema)
    - Defines a generic network model and results representation and many basic discipline independent data structures – may contain lumped parameter as well as FE, FV definitions
    - Can be used for analysis, test and operation models
  - The TAS (Thermal Analysis for Space) protocol (with EXPRESS schema)
    - Adds specific data structures for space thermal analysis
  - The runtime-loaded TAS Dictionary
    - Defines a large set of standard NRF and TAS instances (units, quantity types, node classes, …)
    - Can be extended in a backwards compatible way without affecting the NRF or TAS protocol or already implemented software

Short history of STEP-TAS development (2)

1991-1993	Precursor: French SET-ATS standard – Some limited implementation in THERMICA and ESARAD
1994	Initial ideas for STEP standard for exchange of thermal models (from ESA ICETAS study)
1995-1997	Development of STEP-NRF and STEP-TAS version 1 Software library by Simulog (France) on top of ST-Developer toolkit by STEP Tools Inc. (USA)
1998	Prototype implementations of STEP-TAS v1 in Europe and US
1999	Implementation of STEP-TAS v1 in industrial releases of ESARAD, THERMICA and Thermal Desktop Not successful: very slow, excessive memory usage and problems with larger models
End 2002-now	Significant simplification of STEP-NRF and STEP-TAS at ESTEC leading to version 2 Development of pyExpress compiler/code generator to remove dependency on COTS toolkits Development of TASverter in Python programming language using library generated by pyExpress Readers & writers for ESARAD, THERMICA and Coratherm – successfully used in industry from August 2003 Start of STEP-SPE (Space Environment analysis model exchange) extension of STEP-TAS Start of formal ECSS and ISO standardisation (preparation of paperwork) Start of full open source STEP development toolkit by University of Manchester (nickname “PyJex”)

STEP-TAS data exchange infrastructure



Main characteristics STEP-TAS (1) "Thermal Analysis for Space"

- “STEP-TAS” is the standard that end-users need be aware of
  - STEP-TAS includes STEP-NRF which is a discipline independent building block
    - NRF provides the general features to enable multi-discipline data exchange
    - NRF enables proper modular software engineering
- Supports three kinds of models:
  - Thermal geometric models represented by bounded surfaces
    - Thin shells with oriented faces, mesh and notional thickness
  - Thermal lumped parameter network models
    - With all typical ESATAN or SINDA like data
  - Thermal test (or flight) models with sensor identification and possible location
    - Represents test article with thermo-couples, thermistors, data acquisition channels, …
    - Can be used in conjunction with corresponding STEP AP203/AP214 CAD model

Main characteristics STEP-TAS (2) "Thermal Analysis for Space"

- Geometric and mathematical submodels – no limitation on depth
- Separate specification of model and (load/analysis/test) case definition
  - Supports multiple case definitions per model
- Arbitrary number and depth of coordinate system transformations
  - Retains human-understandable rotations – sequence of rotations w.r.t. the major axes
- Mesh definitions on geometric faces
- Mapping from geometric faces to thermal mathematical model nodes
- Rigid body kinematics with on-orbit pointing for articulated parts

Main characteristics STEP-TAS (3) "Thermal Analysis for Space"

- Space trajectory, attitude and orientation
  - Keplerian or general ephemeris orbit arc definition
  - Support for definition of discrete events, sequencing of cases, parameterized attitude, etc.
- Named materials with their thermo-optical and physical properties
  - Supports multiple sets of properties with material property environment (e.g. BOL, EOL)
- Analysis, test or operation results with complete run-execution information
  - Date & time stamp of execution start and end, tool/facility name and version, etc.
- Supports choice of SI or other unit systems (but requires one consistent set)
  - Conversion factors and offsets w.r.t. SI reference units are explicitly defined
  - STEP-TAS dictionary fully defines all Imperial units used in US projects

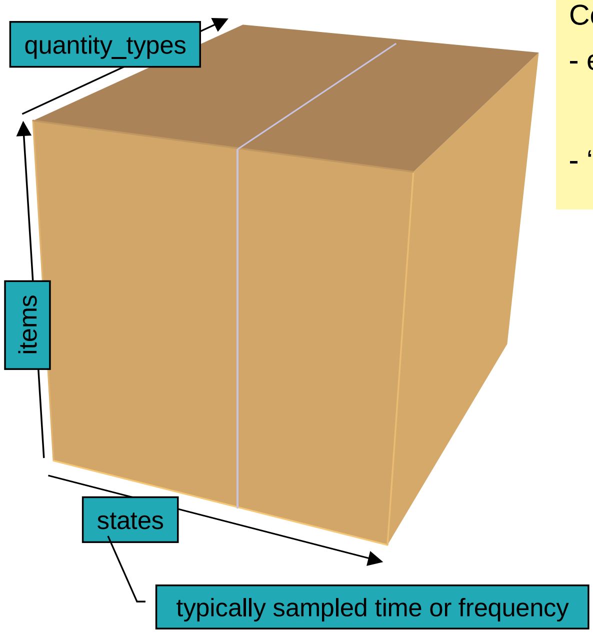
Main characteristics STEP-TAS (4) "Thermal Analysis for Space"

- A 'Conformance Class' is a consistent subset of a STEP protocol
  - A STEP-compliant import/export interface is required to implement complete Conformance Classes
- STEP-TAS Conformance Classes:
  - CC-1: Thermal radiation and conduction model defined by shell geometry
  - CC-2: CC-1 plus kinematic model
  - CC-3: CC-1 plus constructive geometry
  - CC-4: CC-3 plus kinematic model
  - CC-5: CC-1 plus space mission aspects
  - CC-6: CC-4 plus space mission aspects
  - CC-7: Thermal lumped parameter model
  - CC-8: CC-7 plus results
  - CC-9: Thermal test or operation model with results

Main characteristics STEP-NRF (1) "Network-model Results Format"

- Generic, discipline-independent protocol to exchange models, cases & results
  - Model definition, using a discrete network representation
    - Supports model/submodel hierarchy (no limitation on depth)
  - Results data, produced in analysis, test or operation
  - Meta-data, which records details of actual analysis, test or operation performed
  - Provides common basis for a suite of multi-discipline exchange standards
- Discipline-dependent data is defined in a runtime-loaded dictionary
- Supports discrete observations: Sampled results at discrete locations for discrete states
  - No support for continuous fields, etc.
- Any quantity has explicit an quantity type and unit – no 'loose' numerical values
  - e.g. quantity type = temperature / unit = kelvin
- Data model designed to cope efficiently with large amounts of results data
  - Built-in support for scalar, vector, matrix, tensor data structures
  - Designed to map well onto existing scientific data storage standards like HDF5

Main characteristics 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 combination of (item, quantity_type, state)
- 'literal' and 'prescription' SUBTYPEs, for literal and generalised functionally prescribed values

```
ENTITY nrf_datacube
  ABSTRACT SUPERTYPE OF( ONEOF(
    nrf_literal_datacube,
    nrf_prescription_datacube ) );
  name              : nrf_label;
  security_class    : OPTIONAL nrf_security_classification_level;
  value_order       : nrf_datacube_order_type;
  quantity_basis    : nrf_quantity_type_list;
  item_basis        : nrf_observable_item_list;
  state_basis       : nrf_state_value_list;
  real_values       : LIST OF REAL;
  integer_values    : LIST OF INTEGER;
WHERE
  wr1: SIZEOF(real_values) = quantity_basis.number_of_real_values
      * SIZEOF(item_basis.items) * SIZEOF(state_basis.state_values);
  wr2: SIZEOF(integer_values) = quantity_basis.number_of_integer_values
      * SIZEOF(item_basis.items) * SIZEOF(state_basis.state_values);
  wr3: nrf_valid_values_in_datacube(SELF);
END_ENTITY;
```


Excerpts from the STEP-TAS dictionary (Example of an NRF dictionary)

ISO 10303-21 version (STEP file)

```
ISO-10303-21;
HEADER;
/* This STEP / ISO 10303-21 file was produced using the pyExpress toolkit */
/* The pyExpress toolkit is developed by the European Space Agency (ESA) */
...
#31=NRF_DIMENSIONAL_EXPONENTS(2.0,1.0,-2.0,0.0,0.0,0.0,0.0);
#32=NRF_DIMENSIONAL_EXPONENTS(-1.0,1.0,-2.0,0.0,0.0,0.0,0.0);
#33=NRF_DIMENSIONAL_EXPONENTS(2.0,0.0,-2.0,0.0,0.0,0.0,0.0);
#34=NRF_DIMENSIONAL_EXPONENTS(0.0,1.0,-1.0,0.0,0.0,0.0,0.0);
#35=NRF_DIMENSIONAL_EXPONENTS(3.0,0.0,-1.0,0.0,0.0,0.0,0.0);
#36=NRF_DIMENSIONAL_EXPONENTS(1.0,-1.0,2.0,0.0,0.0,0.0,0.0);
#37=NRF_SI_UNIT(*,*,*,'','metre',*);
#38=NRF_SI_UNIT(*,*,*,'kilo','gram',*);
#39=NRF_SI_UNIT(*,*,*,'','second',*);
#40=NRF_SI_UNIT(*,*,*,'','ampere',*);
#41=NRF_SI_UNIT(*,*,*,'','kelvin',*);
#42=NRF_SI_UNIT(*,*,*,'','radian',*);
#43=NRF_SI_UNIT(*,*,*,'','degree_Celsius',*);
#44=NRF_SI_UNIT(*,*,*,'','newton',*);
#45=NRF_SI_UNIT(*,*,*,'','joule',*);
#46=NRF_SI_UNIT(*,*,*,'','watt',*);
#47=NRF_SI_UNIT(*,*,*,'milli','metre',*);
#48=NRF_SI_UNIT(*,*,*,'centi','metre',*);
#49=NRF_CONVERSION_BASED_UNIT('degree','deg',*,#42,1.74532925199
#50=NRF_DERIVED_UNIT_ELEMENT(#49,1.0);
#51=NRF_DERIVED_UNIT_ELEMENT(#39,-1.0);
#52=NRF_DERIVED_UNIT('degree per second',*,#20,(#50,#51));
#53=NRF_CONTEXT_DEPENDENT_UNIT('dimensionless','-',#19);
#54=NRF_DERIVED_UNIT_ELEMENT(#37,1.0);
#55=NRF_DERIVED_UNIT_ELEMENT(#39,-1.0);
#56=NRF_DERIVED_UNIT('metre per second',*,#24,(#54,#55));
...
#121=NRF_BASIC_QUANTITY_TYPE('t','time','Time',#12);
#122=NRF_BASIC_QUANTITY_TYPE('T','temperature','Temperature',#14
#123=NRF_BASIC_QUANTITY_TYPE('rho','mass_density','Unit mass per
#124=NRF_BASIC_QUANTITY_TYPE('C','specific_heat_capacity','Unit
#125=NRF_QUANTITY_TYPE_QUALIFIER('p','constant_pressure','At con
...
END-ISO-10303-21;
```

HTML version

tas_arm_dictionary - Microsoft Internet Explorer

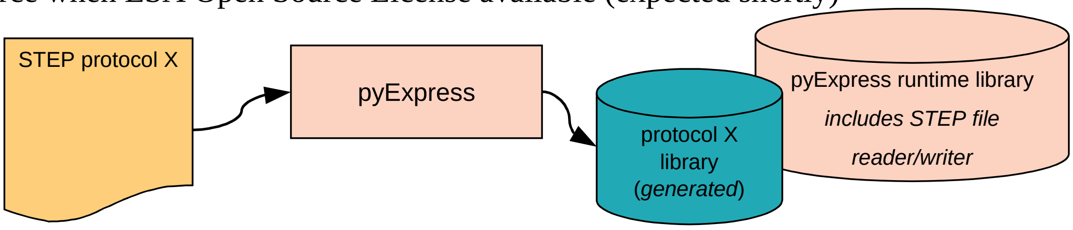
Address C:\Documents and Settings\hanspeter\My Documents\0-sandbox\step-tas\tas_arm_dictionary.html

Dictionary quantityTypesSi: SI-based real quantity types (Nrf_real_quantity_type)

key/qualified name	symbol	base quantity	qualifiers	unit	lowerbound	upperbound
absorbed_albedo power	Q_A	power	absorbed_albedo (A)	watt	NA	NA
absorbed_internal power	Q_I	power	absorbed_internal (I)	watt	NA	NA
absorbed_planet_infra_red power	Q_E	power	absorbed_planet_infra_red (E)	watt	NA	NA
absorbed_rest power	Q_R	power	absorbed_rest (R)	watt	NA	NA
absorbed_solar power	Q_S	power	absorbed_solar (S)	watt	NA	NA
area	A	area	NA	square metre	>=0.0	NA
constant_pressure heat_capacity	mC_p	heat_capacity	constant_pressure (p)	joule per kelvin	>=0.0	NA
constant_pressure specific_heat_capacity	C_p	specific_heat_capacity	constant_pressure (p)	joule per kilogram kelvin	>=0.0	NA
cross_sectional_flow area	A_cf	area	cross_sectional_flow (cf)	square metre	>=0.0	NA
fluid_conductor	GP	fluid_conductor	NA	joule per pascal	>=0.0	NA
hydraulic diameter	D_F	diameter	hydraulic (F)	metre	>=0.0	NA
hydraulic length	L_F	length	hydraulic (F)	metre	>=0.0	NA
incident_albedo power	Q_AI	power	incident_albedo (AI)	watt	NA	NA
incident_planet_infra_red power	Q_EI	power	incident_planet_infra_red (EI)	watt	NA	NA
incident_solar power	Q_SI	power	incident_solar (SI)	watt	NA	NA
infra_red diffuse transmittance	tau_ir_dif	transmittance	infra_red (ir) diffuse (dif)	dimensionless	>=0.0	<=1.0

Implementation software - pyExpress

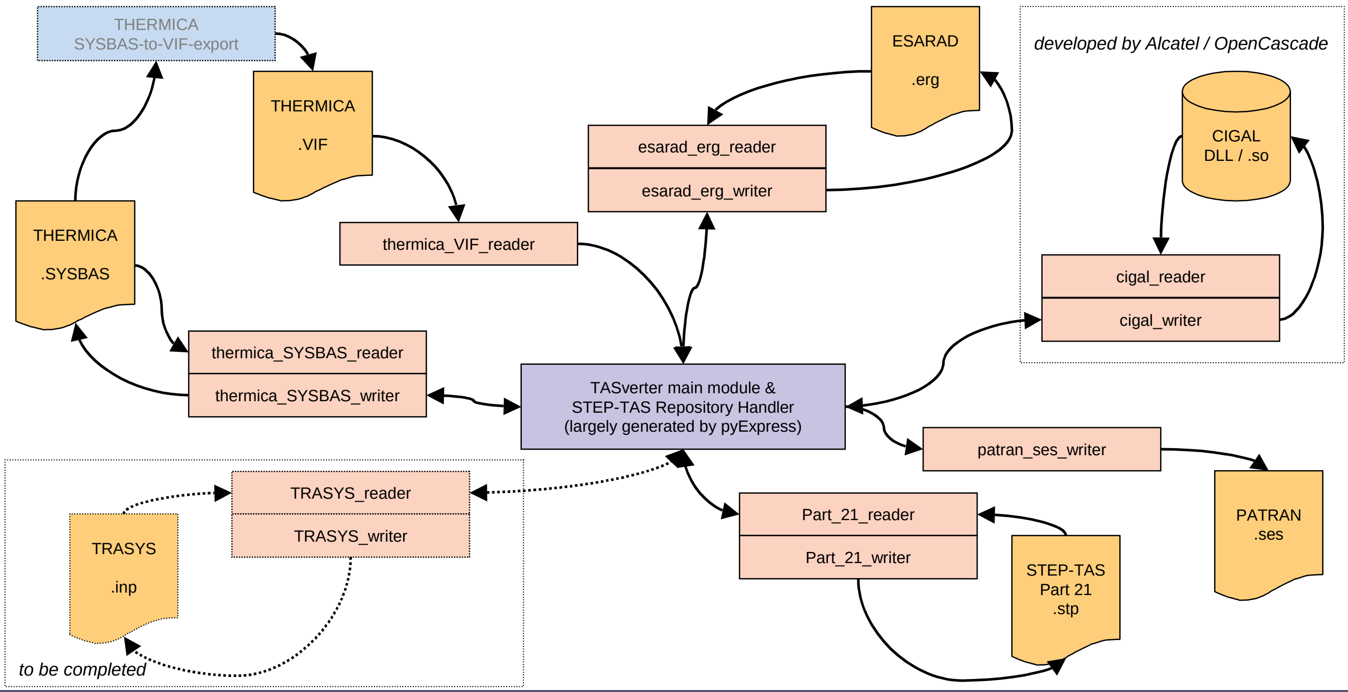
- pyExpress is an EXPRESS compiler / code generator / runtime environment
  - EXPRESS is the STEP data model definition language (ISO 10303-11)
  - Developed by ESA/ESTEC in Python programming/scripting language
    - Python is a freely available, open source, object-oriented language (www.python.org)
    - Very powerful, short development cycle, good performance
    - Very good interfacing with C/C++
      - Use SWIG to generate Python layer on existing C/C++ library (www.swig.org)
    - Provided as open source to ESA contractors – will be made available as global open source when ESA Open Source License available (expected shortly)



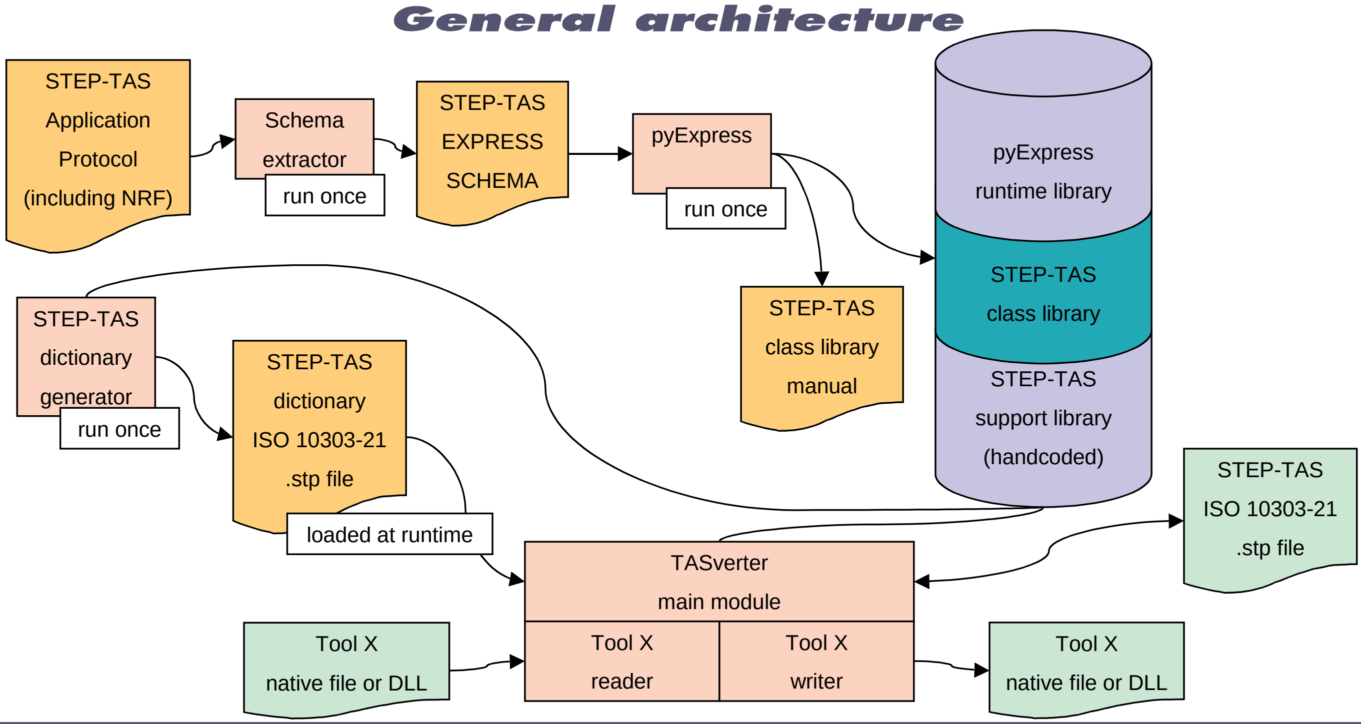
Implementation software – TASverter (1)

- TASverter is a STEP-TAS model conversion tool
- Developed by ESA/ESTEC in Python since January 2003
- Objectives
  - Offer end-users finally a properly working solution for exchange of thermal models
    - First between major European analysis tools ESARAD and THERMICA
  - Produce a fully functional open source framework for STEP-TAS
    - Including extensive validation and verification
  - Create maintainable and cost-effective implementation alternative
    - Can be used by converter developer with minimal STEP knowledge
  - Ensure long term availability, i.e. no dependence on any proprietary software

Implementation software – TASverter (2)



Implementation software – TASverter (3)
General architecture



Implementation software Verification Test Suite

- More than 200 unit tests (CC-1 and CC-3)
  - 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, e.g.:
  - ATV (Automated Transfer Vehicle) model
  - METOP C/D full spacecraft model
  - NASA's ISS thermal interface model
  - Herschel-Planck full spacecraft model
  - Integral full spacecraft model
- All unit tests and most real models (some cannot be made public) will be made available to STEP-TAS interface developers

Schedule (1)

- Freeze of STEP-NRF and STEP-TAS protocols in Nov 2004
- Update of TASverter to support final STEP-TAS standard
  - Release expected Jan 2005 (CC-1 and CC-3)
- Transfer THERMICA reader/writer modules to Astrium SAS for further maintenance
- Prepare and submit NRF and TAS to ECSS and ISO TC 184 / SC 4 for formal standardisation
  - ECSS = European Cooperation for Space Standardization
- Publish standards and software as open source
  - pyExpress and TASverter
  - on ESA website with full configuration control
  - STEP-TAS and STEP-NRF schemas, Python libraries
  - Pending on completion of formal ESA Open Source License (expected 2004-Q3)
- ESATAN model and results writer being developed by ESTEC in frame of ESATAP project
  - First delivery took place 1 Oct 2004 – validated protocol to support ESATAN/SINDA type models
- Upgraded BagheraView – independent STEP-TAS viewer/reporter
  - Development ongoing under CNES contract

Schedule (2)

- ESA funded development of STEP-SPE (Space Environmental Analysis)
  - Start October 2003 – Scheduled for completion in 2005
  - Extends STEP-TAS for micro-meteorites/debris, contamination, atomic oxygen, high energy particle radiation, plume impingement, etc.
- Full open source EXPRESS software development toolkit – nickname ‘PyJex’
  - ESA contract to Computer Science group in University of Manchester
  - Development ongoing and progressing well since April 2004
  - Provide full EXPRESS compiler with open backend / code generators for C/C++, Java and Python
  - Python API will be backward compatible with pyExpress generated API
  - Public release scheduled for April 2005
- Add conformance classes to existing readers / writers:
  - Kinematics and mission aspects (release expected 2005 Q2)
- Promote implementation of STEP-TAS in US and Canadian tools
  - TMG, Thermal Desktop, TSS, ...

Schedule (3)

- New readers and writers
  - Transform existing TRASYS/ESARAD converter to TRASYS/TAS reader/writer
  - Transform existing SINDA85/ESATAN converter to SINDA85/TAS reader
  - Add STEP AP203 reader/writer, with primitive shape recognition capability
    - Can be derived from existing AP203/ESARAD converter plus old TAS version 1 mapping and facetting of remaining NURBS surfaces
  - Mapping to HDF5 in stead of ISO 10303-21 for efficient handling of large datasets

Closing statements

- ESA is fully committed to making STEP-TAS a success
  - Funding and maintaining robust open data exchange standards and software is fully in line with the Agency's mandate
  - It's a key element in Thermal and Space Environment Analysis Software Harmonisation
- The user community as a whole will benefit from reliable STEP-TAS middleware
  - Both end-users and developers
  - Our hope is that it will create a higher level playing field with healthy competition between the analysis tools while still safeguarding the long term interests of end-users

Acknowledgements

- Simon Appel, David Alsina, Duncan Gibson (ESA/ESTEC)
- Olivier Pailles, Arnaud Klinger (Incka, previously Simulog, France)
- Eric Lebègue and co-workers (CSTB/GRAITEC, France)
- Julian Thomas, David Scurrah, John Hurdle (ALSTOM Power, UK)
- Marc Jacquiau (Astrium SAS, France)
- Thierry Basset (Alcatel Space, France)
- Christian Caillet (OpenCascade, France)
- Georg Siebes (NASA-JPL, US)

References

- ISO 10303 (STEP) standards
  http://www.tc184-sc4.org
- STEP-TAS
  http://www.estec.esa.int/thermal/tools/standards.html
- TASverter
  http://www.estec.esa.int/thermal/tools/tasverter.html
- Annual NASA-ESA workshop on product data exchange
  6th edition, 2004 April 20-23, at EADS/Astrium, Friedrichshafen, Germany
  http://www.estec.esa.int/conferences or http://step.jpl.nasa.gov
- Hierarchical Data Format (version 5) – HDF5
  http://hdf.nsca.uiuc.edu
- European Cooperation for Space Standardization
  http://www.ecss.nl
- Python (freely available open source scripting language)
  http://www.python.org
- SWIG (freely available open source programming language interface generator)
  http://www.swig.org