Project
Development of Thermal Neutral Formats
ESA Contract No. 10596/93/NL/FG

STEP Application Protocol

Thermal Analysis for Space

STEP-TAS-012-AP

Version 2

Release 2.4 (28 March 1997)

European Space Agency

French Space Agency

Fokker Space

Association GOSET

GEC ALSTHOM
EUROPEAN GAS TURBINES
MECHANICAL ENGINEERING CENTRE

Abstract

This document is a STEP Application Protocol for the electronic exchange of product model data in the field of Thermal Analysis for Space applications (STEP-TAS). STEP (STandard for the Exchange of Product model data) is the casual name for the ISO 10303 standard. STEP-TAS is a so-called External Application Protocol, which means it is not an official ISO work item. Its development is initiated and funded by the European Space Agency (ESA) in corporation with the French Space Agency (CNES).

The STEP-TAS Application Protocol attempts to address the industrial need for a stable, long term solution for efficient and reliable exchange of space thermal engineering model and results data. The thermal control engineering of spacecraft and other space-related equipment requires extensive computer-supported analyses and tests. In typical space projects several companies and/or institutions work together. They are often located at different sites, which may be geographically far apart and they often use different computer platforms.

ISO 10303 / STEP is a series of International Standards for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also a basis for implementing and sharing product databases and archiving. The development of the ISO 10303 standard is a world-wide undertaking covering all engineering disciplines and all industrial branches. It started in 1984 and is the most comprehensive effort in its kind.

Current status and ongoing activities

The current Release 2.4 is the fifth release for Version 2 of the STEP-TAS Application Protocol.

In this release the reader will find a revised and extended Application Reference Model (ARM), which defines the complete set of information requirements for thermal-radiative analysis for space. The ARM has been updated to be fully harmonised with the [STEP-NRF] protocol: the STEP-TAS protocol is now a strict superset of the [STEP-NRF] protocol. Also the Application Interpreted Model (AIM) is now fully developed to the requirements set by the ARM. This means that the current release is a full implementable Application Protocol.

Colophon

This document was produced as part of the ESA-study “Thermal Neutral Format Based on STEP” (ESA/ESTEC Contract Number 10596/93/NL/FG) with additional funding by the French Space Agency (CNES) for the preparation of Clause 5 and Annexes A, B, G, H.

The study and review team consisted of:

- Hans Peter de Koning, Fokker Space BV, Leiden, The Netherlands – Study manager and author of all parts except Clause 5 and Annexes A, B, G, H.
  (phone: +31-71-5245829, fax: +31-71-5245845, e-mail: h.p.de.koning@fokkerspace.nl)
- Pau Planas Almazan, ESA/ESTEC/YCV, Noordwijk, The Netherlands – ESA study manager and reviewer.
  (phone: +31-71-5654528, fax: +31-71-5656142, e-mail: pau@yc.estec.esa.nl)
- Jean-Luc Le Gal, CNES, Toulouse, France – CNES study manager and reviewer.
- Pascal Huau, Association GOSET, Nanterre (Paris), France – Reviewer and author of Clause 5 and Annexes A, B, G, H.
- Bruno Desaunettes, EPSILON Ingénierie, Labège (Toulouse), France – Reviewer.
- Eric Lebegue, ESPRI Concept, Sophia Antipolis, France – Reviewer.
- Peter Willems, TNO Building and Construction Research, Delft, The Netherlands – Reviewer.
- Christian Puillet, CNES, Toulouse, France – Reviewer.
- Hans de Wolf, Fokker Space BV, Leiden, The Netherlands – Reviewer.

The document was prepared with the FrameMaker®4 document processor.

Document Status Sheet

document title	STEP AP - Thermal Analysis for Space
document reference id	STEP-TAS-012-AP
version, release, date	*reason for change*
v1, r1.0, 25-May-94	Created
v1, r1.1, 24-Jun-94	First version of AAM (Annex E) and ARM (Clause 4) defined.
v1, r1.2, 11-Jul-94	Second version of AAM (Annex E) and ARM (Clause 4). Completed Clause 3 - “Definitions and Abbreviations”. Processed comments recorded in STEP-TAS-011-MM, except addition of AAM example.
v1, r1.3, 25-Jul-94	Updated AAM and ARM according to comments in STEP-TAS-018-MM. Preliminary scope of ARM to be implemented in AIM indicated by √-AIM markers in Clause 4.
v1, r1.4, 23-Sep-94	Updated Clauses 3, 4 and Annex F: • Processing the comments in review documents STEP-TAS-AG-001-TN by GOSET and STEP-TAS-EPSI-002-TN by EPSILON. • Completing the AAM and ARM to provide a stable basis for AIM Version 1 development. Major updates of model structure, orbit and attitude definition.
v1, r1.5, 2-Dec-94	Complete last update of the document before start of AIM development. Comments from STEP-TAS-AG-002-TN and review meetings STEP-TAS-026-MM and STEP-TAS-030-MM were processed.
v1, r1.6, 22-Dec-94	Editorial corrections and inclusion of first working draft of AIM (Clause 5 and Annex A). Corrections according to STEP-TAS-034-MM implemented.
v1, r1.7, 11-May-95	The AIM (Clause 5 and Annexes A, B, G, H) – mapping table & EXPRESS definition – was completed for all in-scope application objects in the ARM. The ARM was updated to reflect the experience gained during AIM development and to make ARM and AIM as much as possible consistent.
v2, r2.0, 29-Jan-96	First Phase 2 upgrade resolving the open item list STEP-TAS-058-TN drawn up at the close of Phase 1.
v2, r2.1, 27-Jun-96	ARM (Clause 4 and Annex F) revised and completed and general updates to Clauses 1, 2 and 3 in line with the remarks from the Release 2.0 Review Meeting held 15-Feb-96, see STEP-TAS-060-MM. The AIM remains at Release 1.7 status and is now out-of-date.
v2, r2.2, 29-Nov-96	Annexes were renumbered in line with the new [STEP-AP-Guide] and the [STEP-NRF-AP] protocol. ARM (Clause 4 and Annex G) harmonized with the STEP-NRF protocol. Furthermore all outstanding issues – mainly raised by P. Huau of GOSET when doing the mapping of the STEP-NRF protocol – were resolved. The STEP-TAS ARM is now a pure extension of the STEP-NRF ARM. The AIM still remains at Release 1.7 status and is still out-of-date.
v2, r2.3, 15-Feb-97	Update of clauses 2 and 4.2 Development of clause 5 and annexes A, B, C, K.

document title	STEP AP - Thermal Analysis for Space
document reference id	STEP-TAS-012-AP
version, release, date	*reason for change*
v1, r1.0, 25-May-94	Created
v1, r1.1, 24-Jun-94	First version of AAM (Annex E) and ARM (Clause 4) defined.
v1, r1.2, 11-Jul-94	Second version of AAM (Annex E) and ARM (Clause 4). Completed Clause 3 - “Definitions and Abbreviations”. Processed comments recorded in STEP-TAS-011-MM, except addition of AAM example.
v1, r1.3, 25-Jul-94	Updated AAM and ARM according to comments in STEP-TAS-018-MM. Preliminary scope of ARM to be implemented in AIM indicated by √-AIM markers in Clause 4.
v1, r1.4, 23-Sep-94	Updated Clauses 3, 4 and Annex F: • Processing the comments in review documents STEP-TAS-AG-001-TN by GOSET and STEP-TAS-EPSI-002-TN by EPSILON. • Completing the AAM and ARM to provide a stable basis for AIM Version 1 development. Major updates of model structure, orbit and attitude definition.
v1, r1.5, 2-Dec-94	Complete last update of the document before start of AIM development. Comments from STEP-TAS-AG-002-TN and review meetings STEP-TAS-026-MM and STEP-TAS-030-MM were processed.
v1, r1.6, 22-Dec-94	Editorial corrections and inclusion of first working draft of AIM (Clause 5 and Annex A). Corrections according to STEP-TAS-034-MM implemented.
v1, r1.7, 11-May-95	The AIM (Clause 5 and Annexes A, B, G, H) – mapping table & EXPRESS definition – was completed for all in-scope application objects in the ARM. The ARM was updated to reflect the experience gained during AIM development and to make ARM and AIM as much as possible consistent.
v2, r2.0, 29-Jan-96	First Phase 2 upgrade resolving the open item list STEP-TAS-058-TN drawn up at the close of Phase 1.
v2, r2.1, 27-Jun-96	ARM (Clause 4 and Annex F) revised and completed and general updates to Clauses 1, 2 and 3 in line with the remarks from the Release 2.0 Review Meeting held 15-Feb-96, see STEP-TAS-060-MM. The AIM remains at Release 1.7 status and is now out-of-date.
v2, r2.2, 29-Nov-96	Annexes were renumbered in line with the new [STEP-AP-Guide] and the [STEP-NRF-AP] protocol. ARM (Clause 4 and Annex G) harmonized with the STEP-NRF protocol. Furthermore all outstanding issues – mainly raised by P. Huau of GOSET when doing the mapping of the STEP-NRF protocol – were resolved. The STEP-TAS ARM is now a pure extension of the STEP-NRF ARM. The AIM still remains at Release 1.7 status and is still out-of-date.
v2, r2.3, 15-Feb-97	Update of clauses 2 and 4.2 Development of clause 5 and annexes A, B, C, K.

document title	STEP AP - Thermal Analysis for Space
document reference id	STEP-TAS-012-AP
version, release, date	*reason for change*
v2, r2.4, 28-Mar-97	Updates of clause 4 on kinematic_model UoF. Update of Clause 5 and Annex A (AIM) to reflect changes in Clause 4. Added definition of conformance classes in Clause 6. Added specification of normative names for classes and roles of valued properties in Annex C. Addition of the full TAS ARM EXPRESS schema in Annex G.

Changes with respect to the previous release are indicated by vertical bars in the margin.

Document Change Record

DCR number		
date		28 March 1997
originator		P. Huau
approved by		
document title		STEP AP - Thermal Analysis for Space
document reference id		STEP-TAS-012-AP
document version/release		Version 2 Release 2.4
page	*section*	*reason for change*

4-2	4.1.2	Removal of modelling_assumption and of modelling_assumption_ relationship.
4-3	4.1.3	Removal of tensor_mask, tensor_mask_member, tensor_mask_name
4-4	4.1.8	Removal of the UoF product_requirement. Removal of clause 4.2.8.
4-5	4.1.11	Move kinematic_articulation in UoF space_mission_aspect.
4-7	4.2	Addition of text to explain where the definitions of the NRF entities are
4-7	4.2.8	Removal of the clause associated with the product_requirements UOF
4-12	4.2.8.10	Addition of “of of_case” at the end of the 2nd assertion
4-16	4.2.8.15	Modification of the attribute radius to enable reference to a property_ function_association
4-17	4.2.8.17	Modification of the attribute black_body_surface_temperature to enable reference to a property_function_association. Correction of the definition of the attribte notional_distance_ ...
4-18	4.2.8.18	Complement added in the definition. Modification of the attribute albed_reflection_factor to enable a reference to a property_function_association. Addition of an assertion about the range of the albedo_reflection_factor.
4-18	4.2.8.19	Complement added in the definition about the local coordinate system.
4-19	4.2.8.20	Correction in the definition about the axis of rotation. Correction of the attribute rotation_axis (reference to an axis_ placement)

4-19	4.2.8.21	Modification of the 2nd sentence of the definition. Correction of the attribute rotation_axis (reference to an axis_ placement). Improvement of the definition of the attribute number_of_spin_ positions.
4-21	4.2.8.23	Removal of the attribute star_name and of the associated assertion. Addition of the attribute star_centre referring to the centre of the considered star.
4-21	4.2.8.25	Modification of the attribute angle to enable a reference to a property_ FUNCTION_association.
4-22	4.2.8.27	Modification of the attribute values to refer to a property_function_ association.
4-23	4.2.9.1	Coorection of the attribute length_unit and plance_angle_unit.
4-23	4.2.9.3	Change the name of the attribute *name* to *role*.
4-24	4.2.9.5	Change of the type of the attribute location.
4-25	4.2.9.7.1	Rewording of the definition of the attribute placement.
4-36	4.2.9.9.1	Rewording of the definiton. Correction of the definition of sense.
4-36	4.2.9.9.2	Move placement information from a note to the normative definition.
4-36	4.2.9.9.3	Move placement information from a note to the normative definition.
4-36	4.2.9.9.4	Creation of the object in order to correct the definition of second_ operand in boolean_difference_surface.
4-38	4.2.9.10.2	Rewording of the definition ogf placement in order to explicit its real semantics.
4-38	4.2.9.10.2	Modification of the attribute second_operand.
4-40	4.2.10.7	Renaming of the object into implicit_revolute_kinematic_pair_value in order to explicit its real semantics (note: the renaming has been forgotten in the first line of the definition)
4-42	4.2.10.10	Addition of an assertion
4-42	4.2.11.2	Simplification of the textual definition.
4-47	4.2.16.3	Removal of the attribute values. Addition of the attribute property_value referring to property_function_ association or property_string_association or property_value_ association. Addition of an assertion about the attribute in_environment.
4-48	4.2.17.2	Improvement of the specification of the attributes factor and offset.
4-48	4.2.17.3	Addition of the object to enable the association of a value (here a index of function) to a property_class_scalar_functional.
4-48	4.2.17.3	Addition of the object to enable the association of a value (here a index of string) to a property_class_scalar_descriptive.
4-48	4.2.17.3	Addition of the object to enable the association of a value to a property_ class_scalar_quantitative.
4-51	4.3	removal of the assertion about geometric_context (was wrong). Addition of an assertion about material.

5-1	5.1	Update of the paragraph 5.1 presenting the mapping syntax. Addition of the list of rules associated with the mappings. Creation of the mapping tables in an annex text file. Note: during the mapping, we found the following problems: • the measure with units of clause 4.2.17 are not consistent with the definition of material_property and cannot be referred to. In consequence, they were not mapped; • the object thermal_radiation_spectrum refers to the object source_ reference which does not exist. (these issues come in addition of the issues raised before January 22 on clause 4.2)
5-x	5.2	Creation of the AIM short form. Creation of the AIM long form (annex A) Creation of the list of short names (annex B). Update of annex C.

Table of Contents

Abstract....ii

Document Status Sheet....iii

Document Change Record....vi

Table of Contents....ix

Foreword....xxi

Introduction....xxii

0.1 Brief introduction on the STEP standard....xxii
0.1.1 The objectives of STEP....xxii
0.1.2 Principles....xxii
0.1.3 The STEP architecture....xxii
0.1.4 The basis of the STEP product data model....xxvi
0.1.5 Realisation of a STEP solution for product data exchange....xxviii
0.1.6 Status of the STEP standard and its implementations....xxviii
0.2 The Industrial Need for STEP -based Data Exchange for Thermal Analysis for Space....xxviii

Clause 1 - Scope....1-1

Clause 2 - Normative References....2-1

Clause 3 - Definitions and Abbreviations....3-1

3.1 Terms Defined in ISO 10303-1....3-1
3.2 Terms Defined in ISO 10303-31....3-2
3.3 Terms Defined in ISO 10303-42....3-2
3.4 Terms Defined in ISO 10303-45....3-2
3.5 Terms Defined in ISO 10303-105....3-3
3.6 Terms Defined in this AP....3-3
3.7 Abbreviations....3-5

Clause 4 - Information Requirements....4-1

4.1 Units of Functionality....4-1
4.1.1 product_structure UoF....4-1
4.1.2 network_model_representation UoF....4-2
4.1.3 bulk_results UoF....4-2
4.1.4 parameterized_functions UoF....4-3
4.1.5 physical_quantities_and_units UoF....4-3
4.1.6 date_and_time UoF....4-3
4.1.7 general_support UoF....4-3
4.1.8 space_mission_aspects UoF....4-4

4.1.9 geometric_model UoF 4-4
4.1.10 kinematic_model UoF 4-5
4.1.11 thermal_radiative_model UoF 4-5
4.1.12 thermal_lumped_parameter_model UoF 4-5
4.1.13 thermal_finite_element_model UoF 4-5
4.1.14 visual_presentation UoF 4-5
4.1.15 material_properties UoF 4-6
4.1.16 additional_physical_quantities_and_measures UoF 4-6
4.2 Application Objects 4-7
4.2.1 product_structure UoF 4-7
4.2.2 network_model_representation UoF 4-7
4.2.3 bulk_results UoF 4-7
4.2.4 parameterized_functions UoF 4-7
4.2.5 physical_quantities_and_units UoF 4-7
4.2.6 date_and_time UoF 4-7
4.2.7 general_support UoF 4-7
4.2.8 space_mission_aspects UoF 4-7
4.2.8.1 space_mission 4-7
4.2.8.2 space_mission_case 4-7
4.2.8.3 space_mission_phase 4-8
4.2.8.4 space_mission_aspect 4-9
4.2.8.5 space_mission_event 4-9
4.2.8.6 space_mission_event_with_epoch 4-9
4.2.8.7 space_mission_event_with_running_time 4-10
4.2.8.8 space_mission_event_with_true_anomaly 4-10
4.2.8.9 space_coordinate_system 4-10
4.2.8.10 orbit_arc 4-11
4.2.8.11 discretised_orbit_arc 4-13
4.2.8.12 keplerian_orbit_arc 4-13
4.2.8.13 kepler_parameter_set 4-13
4.2.8.14 orbit_position_and_velocity 4-16
4.2.8.15 celestial_body 4-16
4.2.8.16 celestial_body_with_orbit 4-17
4.2.8.17 thermal_aspects_of_celestial_body 4-17
4.2.8.18 albedo_reflector 4-18
4.2.8.19 main_body_placement 4-18
4.2.8.20 main_body_placement_with_constant_rotation 4-19
4.2.8.21 main_body_placement_with_spinning 4-19
4.2.8.22 pointing_in_space 4-20
4.2.8.23 general_star_pointing 4-21
4.2.8.24 general_space_pointing 4-21
4.2.8.25 attitude_rotation 4-21

4.2.8.26 kinematic_articulation 4-22
4.2.8.27 thermal_environment 4-22
4.2.8.28 thermal_radiation_spectrum 4-23
4.2.8.29 operational_mode 4-23
4.2.8.30 degradation_dose_profile 4-23
4.2.9 geometric_model UoF 4-23
4.2.9.1 geometric_context 4-23
4.2.9.2 geometric_model 4-23
4.2.9.3 geometric_submodel_usage 4-24
4.2.9.4 geometric_construction_point 4-24
4.2.9.5 axis_placement 4-24
4.2.9.6 Surface definitions 4-25
4.2.9.7 Primitive bounded surfaces 4-25
4.2.9.8 Compound bounded surfaces 4-35
4.2.9.9 Auxiliary boolean construction surfaces 4-36
4.2.9.10 Boolean result surfaces 4-38
4.2.10 kinematic_model UoF 4-38
4.2.10.1 kinematic_model 4-38
4.2.10.2 kinematic_link 4-40
4.2.10.3 kinematic_joint 4-41
4.2.10.4 kinematic_pair 4-41
4.2.10.5 kinematic_pair_value 4-41
4.2.10.6 revolute_kinematic_pair 4-41
4.2.10.7 explicit_revolute_kinematic_pair_value 4-42
4.2.10.8 implicit_revolute_kinematic_pair_value 4-42
4.2.10.9 prismatic_kinematic_pair 4-43
4.2.10.10 prismatic_kinematic_pair_value 4-43
4.2.11 thermal_radiative_model UoF 4-43
4.2.11.1 basic_thermal_radiative_model 4-43
4.2.11.2 thermal_radiative_face_set 4-44
4.2.11.3 meshed_thermal_radiative_model 4-44
4.2.11.4 thermal_radiative_meshing 4-45
4.2.11.5 thermal_radiative_meshing_with_incremental_node_list 4-46
4.2.11.6 thermal_radiative_meshing_with_full_node_list 4-46
4.2.11.7 enclosure 4-46
4.2.12 thermal_lumped_parameter_model UoF 4-46
4.2.13 thermal_finite_element_model UoF 4-47
4.2.14 visual_presentation UoF 4-47
4.2.14.1 colour_rgb 4-47
4.2.15 material_properties UoF 4-47
4.2.15.1 material 4-47
4.2.15.2 product_material_relationship 4-47

4.2.15.3 material_property 4-48
4.2.15.4 material_property_environment 4-48
4.2.15.5 commercial_aspects_of_material 4-48
4.2.16 additional_physical_quantities_and_measures UoF 4-48
4.2.16.1 derived_unit 4-48
4.2.16.2 conversion_unit 4-49
4.2.16.3 function_valued_property 4-49
4.2.16.4 string_valued_property 4-49
4.2.16.5 number_valued_property 4-49
4.3 Application Assertions 4-50

Clause 5 - Application Interpreted Model 5-1
5.1 Mapping Table 5-1
5.1.1 Mapping Table - product_structure UoF 5-2
5.1.1.1 ARM_OBJECT product 5-2
5.1.1.2 ARM_OBJECT product_context 5-2
5.1.1.3 ARM_OBJECT product_definition 5-3
5.1.1.4 ARM_OBJECT product_definition_context 5-3
5.1.1.5 ARM_OBJECT product_next_assembly_usage_relationship 5-3
5.1.1.6 ARM_OBJECT product_version 5-3
5.1.2 Mapping Table - network_model_representation UoF 5-4
5.1.2.1 ARM_OBJECT model_represents_product_relationship 5-4
5.1.2.2 ARM_OBJECT network_model 5-4
5.1.2.3 ARM_OBJECT network_node 5-5
5.1.2.4 ARM_OBJECT network_node_class 5-6
5.1.2.5 ARM_OBJECT network_node_relationship 5-6
5.1.2.6 ARM_OBJECT network_node_relationship_class 5-7
5.1.2.7 ARM_OBJECT network_node_relationship_usage 5-7
5.1.2.8 ARM_OBJECT network_node_usage 5-7
5.1.2.9 ARM_OBJECT submodel_usage 5-7
5.1.3 Mapping Table - bulk_results UoF 5-8
5.1.3.1 ARM_OBJECT aspect 5-8
5.1.3.2 ARM_OBJECT ato_campaign 5-8
5.1.3.3 ARM_OBJECT ato_case 5-10
5.1.3.4 ARM_OBJECT ato_phase 5-11
5.1.3.5 ARM_OBJECT list_of_descriptive_values 5-12
5.1.3.6 ARM_OBJECT list_of_functions 5-12
5.1.3.7 ARM_OBJECT model_component_sequence 5-12
5.1.3.8 ARM_OBJECT property_class 5-13
5.1.3.9 ARM_OBJECT property_class_scalar 5-13
5.1.3.10 ARM_OBJECT property_class_scalar_descriptive 5-13
5.1.3.11 ARM_OBJECT property_class_scalar_functional 5-13
5.1.3.12 ARM_OBJECT property_class_scalar_quantitative 5-13

5.1.3.13 ARM_OBJECT property_class_tensor 5-13
5.1.3.14 ARM_OBJECT property_class_usage 5-13
5.1.3.15 ARM_OBJECT property_name 5-14
5.1.3.16 ARM_OBJECT property_role 5-14
5.1.3.17 ARM_OBJECT run 5-14
5.1.3.18 ARM_OBJECT run_sequence 5-15
5.1.3.19 ARM_OBJECT scalar_in_tensor 5-16
5.1.3.20 ARM_OBJECT scan 5-16
5.1.3.21 ARM_OBJECT scan_derivation_procedure 5-17
5.1.3.22 ARM_OBJECT scan_mask 5-17
5.1.3.23 ARM_OBJECT scan_of_derived_values 5-17
5.1.3.24 ARM_OBJECT scan_of_sampled_values 5-18
5.1.3.25 ARM_OBJECT unit_assignment 5-18
5.1.4 Mapping Table - parameterized_function UoF 5-18
5.1.4.1 ARM_OBJECT cyclic_tabular_function 5-18
5.1.4.2 ARM_OBJECT parameterized_function 5-18
5.1.4.3 ARM_OBJECT parameterized_function_with_formula 5-19
5.1.4.4 ARM_OBJECT polynomial_function 5-19
5.1.4.5 ARM_OBJECT tabular_function 5-20
5.1.5 Mapping Table - date_and_time UoF 5-20
5.1.5.1 ARM_OBJECT calendar_date 5-20
5.1.5.2 ARM_OBJECT coordinated_universal_time_offset 5-20
5.1.5.3 ARM_OBJECT date 5-20
5.1.5.4 ARM_OBJECT date_and_time 5-20
5.1.5.5 ARM_OBJECT local_time 5-20
5.1.6 Mapping Table - general_support UoF 5-21
5.1.6.1 ARM_OBJECT address 5-21
5.1.6.2 ARM_OBJECT approval 5-21
5.1.6.3 ARM_OBJECT document 5-21
5.1.6.4 ARM_OBJECT document_type 5-21
5.1.6.5 ARM_OBJECT document_usage_constraint 5-21
5.1.6.6 ARM_OBJECT organization 5-21
5.1.6.7 ARM_OBJECT organization_address 5-21
5.1.6.8 ARM_OBJECT organizational_project 5-21
5.1.6.9 ARM_OBJECT person 5-22
5.1.6.10 ARM_OBJECT person_and_organization 5-22
5.1.6.11 ARM_OBJECT personal_address 5-22
5.1.6.12 ARM_OBJECT security_classification_level 5-22
5.1.7 Mapping Table - space_mission_aspects UoF 5-22
5.1.7.1 ARM_OBJECT space_mission 5-22
5.1.7.2 ARM_OBJECT space_mission_case 5-22
5.1.7.3 ARM_OBJECT space_mission_phase 5-22

5.1.7.4 ARM_OBJECT space_mission_aspect 5-23
5.1.7.5 ARM_OBJECT space_mission_event 5-24
5.1.7.6 ARM_OBJECT space_mission_event_with_epoch 5-24
5.1.7.7 ARM_OBJECT space_mission_event_with_running_time 5-24
5.1.7.8 ARM_OBJECT space_mission_event_with_true_anomaly 5-24
5.1.7.9 ARM_OBJECT space_coordinate_system 5-24
5.1.7.10 ARM_OBJECT orbit_arc 5-25
5.1.7.11 ARM_OBJECT discretised_orbit_arc 5-26
5.1.7.12 ARM_OBJECT keplerian_orbit_arc 5-27
5.1.7.13 ARM_OBJECT kepler_parameter_set 5-27
5.1.7.14 ARM_OBJECT orbit_position_and_velocity 5-28
5.1.7.15 ARM_OBJECT celestial_body 5-28
5.1.7.16 ARM_OBJECT celestial_body_with_orbit 5-29
5.1.7.17 ARM_OBJECT thermal_aspects_of_celestial_body 5-29
5.1.7.18 ARM_OBJECT albedo_reflector 5-30
5.1.7.19 ARM_OBJECT main_body_placement 5-31
5.1.7.20 ARM_OBJECT main_body_placement_with_constant_rotation 5-31
5.1.7.21 ARM_OBJECT main_body_placement_with_spinning 5-32
5.1.7.22 ARM_OBJECT pointing_in_space 5-32
5.1.7.23 ARM_OBJECT general_star_pointing 5-33
5.1.7.24 ARM_OBJECT general_space_pointing 5-33
5.1.7.25 ARM_OBJECT attitude_rotation 5-34
5.1.7.26 ARM_OBJECT kinematic_articulation 5-34
5.1.7.27 ARM_OBJECT thermal_environment 5-34
5.1.7.28 ARM_OBJECT thermal_radiation_spectrum 5-35
5.1.8 Mapping Table - geometric_model UoF 5-35
5.1.8.1 ARM_OBJECT geometric_context 5-35
5.1.8.2 ARM_OBJECT geometric_model 5-36
5.1.8.3 ARM_OBJECT geometric_submodel_usage 5-37
5.1.8.4 ARM_OBJECT geometric_construction_point 5-38
5.1.8.5 ARM_OBJECT axis_placement 5-38
5.1.8.6 ARM_OBJECT primitive_bounded_surface 5-38
5.1.8.7 ARM_OBJECT triangle 5-39
5.1.8.8 ARM_OBJECT rectangle 5-40
5.1.8.9 ARM_OBJECT quadrilateral 5-41
5.1.8.10 ARM_OBJECT disc 5-43
5.1.8.11 ARM_OBJECT cylinder 5-44
5.1.8.12 ARM_OBJECT cone 5-46
5.1.8.13 ARM_OBJECT sphere 5-47
5.1.8.14 ARM_OBJECT paraboloid 5-48
5.1.8.15 ARM_OBJECT compound_surface_shell 5-50
5.1.8.16 ARM_OBJECT boolean_construction_surface 5-50

5.1.8.17 ARM_OBJECT half_space_plane 5-51
5.1.8.18 ARM_OBJECT half_space_cylinder 5-51
5.1.8.19 ARM_OBJECT generalized_half_space 5-51
5.1.8.20 ARM_OBJECT boolean_result_surface 5-51
5.1.8.21 ARM_OBJECT boolean_difference_surface 5-52
5.1.9 Mapping Table - kinematic_model UoF 5-53
5.1.9.1 ARM_OBJECT kinematic_model 5-53
5.1.9.2 ARM_OBJECT kinematic_link 5-53
5.1.9.3 ARM_OBJECT kinematic_joint 5-54
5.1.9.4 ARM_OBJECT kinematic_pair 5-54
5.1.9.5 ARM_OBJECT kinematic_pair_value 5-54
5.1.9.6 ARM_OBJECT revolute_kinematic_pair 5-54
5.1.9.7 ARM_OBJECT explicit_revolute_kinematic_pair_value 5-55
5.1.9.8 ARM_OBJECT implicit_revolute_kinematic_pair_value 5-55
5.1.9.9 ARM_OBJECT prismatic_kinematic_pair 5-55
5.1.9.10 ARM_OBJECT prismatic_kinematic_pair_value 5-55
5.1.10 Mapping Table - thermal_radiative_model UoF 5-55
5.1.10.1 ARM_OBJECT basic_thermal_radiative_model 5-55
5.1.10.2 ARM_OBJECT thermal_radiative_face_set 5-56
5.1.10.3 ARM_OBJECT meshed_thermal_radiative_model 5-56
5.1.10.4 ARM_OBJECT thermal_radiative_meshing 5-56
5.1.10.5 ARM_OBJECT thermal_radiative_meshing_with_incremental_node_list 5-56
5.1.10.6 ARM_OBJECT thermal_radiative_meshing_with_full_node_list 5-57
5.1.10.7 ARM_OBJECT enclosure 5-57
5.1.11 Mapping Table - visual_presentation UoF 5-57
5.1.11.1 ARM_OBJECT colour_rgb 5-57
5.1.12 Mapping Table - material_properties UoF 5-57
5.1.12.1 ARM_OBJECT material 5-57
5.1.12.2 ARM_OBJECT product_material_relationship 5-58
5.1.12.3 ARM_OBJECT material_property 5-59
5.1.12.4 ARM_OBJECT material_property_environment 5-59
5.1.12.5 ARM_OBJECT commercial_aspect_of_material 5-59
5.1.13 Mapping Table - additional_physical_quantities_and_measures UoF 5-60
5.1.13.1 ARM_OBJECT derived_unit 5-60
5.1.13.2 ARM_OBJECT conversion_unit 5-60
5.1.13.3 ARM_OBJECT function_valued_property 5-60
5.1.13.4 ARM_OBJECT string_valued_property 5-61
5.1.13.5 ARM_OBJECT number_valued_property 5-61
5.1.14 Global rules for mapping 5-61
5.2 AIM EXPRESS short listing 5-63
5.2.1 Fundamental concepts and choices 5-66
5.2.1.1 Parameters and result of functions 5-66

5.2.1.2 Polynomial functions 5-66
5.2.1.3 Property classes 5-67
5.2.1.4 Relating a product to its various properties 5-67
5.2.1.5 Celestial bodies 5-67
5.2.1.6 Representations 5-67
5.2.1.7 Geometric models 5-67
5.2.1.8 Sub_model occurences 5-68
5.2.1.9 Surfaces 5-68
5.2.1.10 Geometric construction point 5-69
5.2.1.11 Placement of a primitive_bounded_surface or of a boolean_result_surface 5-69
5.2.1.12 Kinematic_model 5-69
5.2.1.13 Kinematic_link 5-69
5.2.1.14 Material 5-70
5.2.1.15 Material_properties 5-70
5.2.1.16 Assignment of units and uncertainties of geometric models 5-70
5.2.1.17 Assignment of units for property classes 5-70
5.2.2 TAS_schema type definitions 5-70
5.2.2.1 security_classified_item 5-70
5.2.2.2 dated_item 5-70
5.2.2.3 approved_item 5-71
5.2.2.4 sourced_item 5-71
5.2.2.5 named_item 5-71
5.2.2.6 documented_item 5-71
5.2.2.7 supplied_item 5-71
5.2.2.8 coloured_item 5-72
5.2.2.9 TAS_radius_of_celestial_body 5-72
5.2.2.10 TAS_black_body_surface_temperature 5-72
5.2.2.11 TAS_albedo_reflection_factor 5-73
5.2.2.12 TAS_property_value 5-74
5.2.3 TAS_schema entity definitions 5-74
5.2.3.1 TAS_security_assignment 5-74
5.2.3.2 TAS_date_assignment 5-74
5.2.3.3 TAS_approval_assignment 5-75
5.2.3.4 TAS_contact_assignment 5-75
5.2.3.5 TAS_name_assignment 5-75
5.2.3.6 TAS_commercial_designation 5-76
5.2.3.7 TAS_supplier 5-76
5.2.3.8 TAS_document_reference 5-76
5.2.3.9 TAS_colour_assignment 5-76
5.2.3.10 TAS_model_product_relationship 5-77
5.2.3.11 NRF_run 5-77
5.2.3.12 NRF_run_phase 5-78

5.2.3.13 NRF_run_results 5-78
5.2.3.14 NRF_run_sequence 5-79
5.2.3.15 NRF_run_sequence_case 5-79
5.2.3.16 NRF_run_in_sequence 5-80
5.2.3.17 TAS_space_coordinate_system 5-80
5.2.3.18 TAS_gravitational_constant 5-81
5.2.3.19 TAS_revolution_period 5-82
5.2.3.20 TAS_notional_distance 5-82
5.2.3.21 TAS_thermal_flux 5-82
5.2.3.22 TAS_thermal_radiation_spectrum 5-83
5.2.3.23 TAS_evaluation_event 5-83
5.2.3.24 TAS_orbit_governor 5-83
5.2.3.25 TAS_orbit_assignment 5-84
5.2.3.26 TAS_orbit_arc 5-84
5.2.3.27 TAS_discretised_orbit_arc 5-85
5.2.3.28 TAS_kepler_orbit_arc 5-86
5.2.3.29 TAS_celestial_body 5-86
5.2.3.30 TAS_celestial_body_with_orbit 5-87
5.2.3.31 TAS_gravitational_aspects_of_celestial_body 5-87
5.2.3.32 TAS_thermal_aspects_of_celestial_body 5-88
5.2.3.33 TAS_albedo_reflector 5-88
5.2.3.34 TAS_albedo_reflection_data 5-89
5.2.3.35 TAS_body_motion_representation 5-90
5.2.3.36 TAS_thermal_environment 5-90
5.2.3.37 TAS_thermal_environment_data 5-91
5.2.3.38 TAS_material_property_environment 5-91
5.2.3.39 TAS_basic_thermal_radiative_model 5-91
5.2.3.40 TAS_meshed_thermal_radiative_model 5-92
5.2.3.41 TAS_enclosure 5-92
5.2.3.42 TAS_material_property 5-92
5.2.4 TAS_schema imported entity modifications 5-93
5.2.4.1 action 5-93
5.2.4.2 action_relationship 5-93
5.2.4.3 action_assignment 5-93
5.2.4.4 application_context 5-93
5.2.4.5 application_protocol_definition 5-93
5.2.4.6 approval 5-94
5.2.4.7 approval_date_time 5-94
5.2.4.8 approval_person_organization 5-94
5.2.4.9 approval_role 5-94
5.2.4.10 approval_status 5-94
5.2.4.11 date 5-94

5.2.4.12 date_time_role 5-95
5.2.4.13 person 5-95
5.2.4.14 person_and_organization_role 5-95
5.2.4.15 security_classification_level 5-95
5.2.4.16 type_qualifier 5-95
5.2.4.17 data_environment 5-95
5.2.4.18 product_context 5-96
5.2.4.19 product_definition_context 5-96
5.2.4.20 product 5-96
5.2.4.21 SIR_product_version 5-96
5.2.4.22 product_definition_formation 5-96
5.2.4.23 product_definition_relationship 5-96
5.2.4.24 product_definition 5-97
5.2.4.25 global_unit_assigned_context 5-97
5.2.4.26 geometric_representation_context 5-97
5.2.4.27 shape_representation 5-97
5.2.4.28 cartesian_point 5-97
5.2.4.29 advanced_face 5-97
5.2.4.30 SIR_thermal_radiative_face_set 5-98
5.2.4.31 SIR_ato_campaign 5-98
5.2.4.32 SIR_model 5-98
5.2.4.33 SIR_node 5-98
5.2.4.34 SIR_node_relationship 5-98
5.2.4.35 SIR_submodel_usage 5-99
5.2.4.36 SIR_aspect 5-99
5.2.4.37 SIR_ato_case 5-99
5.2.4.38 SIR_ato_phase 5-99
5.2.4.39 SIR_component_sequence 5-99
5.2.4.40 SIR_boolean_result_surface 5-99
5.2.4.41 SIR_space_mission_aspect 5-100
5.2.4.42 SIR_kinematic_link 5-100
5.2.4.43 kinematic_link 5-100
5.2.4.44 mechanism 5-100
5.2.5 TAS_schema rule definitions 5-100
5.2.5.1 ato_campaign_requires_creation_date 5-100
5.2.5.2 ato_campaign_requires_contact 5-101
5.2.5.3 at_most_one_modification_date 5-101
5.2.5.4 at_most_one_security_level 5-101
5.2.5.5 application_context_requires_ap_definition 5-102
5.2.5.6 approval_requires_approval_date_time 5-102
5.2.5.7 approval_date_time_requires_date_time 5-103
5.2.5.8 approval_requires_approval_person_organization 5-103

5.2.5.9 approval_person_organization_requires_person_organization 5-104
5.2.5.10 approvals_are_assigned 5-104
5.2.5.11 component_sequence_requires_aspect 5-104
5.2.5.12 dependent_instantiable_approval_role 5-105
5.2.5.13 dependent_instantiable_approval_status 5-105
5.2.5.14 dependent_instantiable_date 5-105
5.2.5.15 dependent_instantiable_date_time_role 5-105
5.2.5.16 dependent_instantiable_person 5-106
5.2.5.17 dependent_instantiable_person_and_organization_role 5-106
5.2.5.18 dependent_instantiable_security_classification_level 5-106
5.2.5.19 dependent_instantiable_type_qualifier 5-107
5.2.5.20 exclusive_subtypes_action_assignment 5-107
5.2.5.21 product_requires_version 5-107
5.2.5.22 model_requires_definition 5-108
5.2.5.23 subtype_mandatory_action 5-108
5.2.5.24 subtype_mandatory_action_relationship 5-108
5.2.5.25 subtype_mandatory_date 5-109
5.2.5.26 subtype_mandatory_product_definition_formation 5-109
5.2.5.27 subtype_mandatory_product_definition_relationship 5-109
5.2.5.28 unique_ids_in_model 5-110
5.2.5.29 product_context_requires_product 5-110
5.2.5.30 prod_def_context_requires_prod_def 5-111
5.2.5.31 restrict_boolean_operator 5-111
5.2.5.32 restrict_mission_aspects 5-111
5.2.5.33 subtype_mandatory_data_environment 5-112
5.2.5.34 TAS_global_units 5-112
5.2.5.35 TAS_dimensionality_is_3 5-113
5.2.5.36 TAS_point_id_unique 5-113
5.2.5.37 at_most_2_faces_per_surface 5-114
5.2.5.38 thermal_face_requires_thermal_model 5-114
5.2.5.39 kinematic_link_requires_shape 5-114
5.2.5.40 restrict_represented_product 5-115
5.2.5.41 subtype_mandatory_kinematic_link 5-115
5.2.5.42 mechanism_requires_ground_representation 5-116
5.2.6 TAS_schema function definitions 5-116
5.2.6.1 get_names 5-116
5.2.6.2 get_tools 5-117
5.2.6.3 get_timestamps 5-117
5.2.6.4 unique_ids 5-118
5.2.6.5 unique_names 5-118
5.2.6.6 valid_properties_of_celestial_body 5-118
5.2.6.7 orbit_assigned_to 5-119

5.2.6.8 TAS_valid_associated_kepler_set 5-120
5.2.6.9 TAS_valid_albedo_reflector 5-120

Clause 6 - Conformance Requirements 6-1

Annex A - AIM EXPRESS expanded listing (normative) A-1
A.1 Introduction A-1
A.2 AIM expanded listing A-1

Annex B - AIM Short Names for Entities (Normative) B-1

Annex C - Implementation Method Specific Requirements (Normative) C-1
C.1 Reference to the AIM schema C-1
C.2 Instance of the entity application_protocol_definition C-1
C.3 Normative names for classes and roles of valued properties C-1

Annex D - Protocol Implementation Conformance Statement (PICS) proforma (Normative) ... D-1

Annex E - TBD (Normative) E-1

Annex F - Application Activity Model (Informative) F-1
F.1 Application Activity Model Definitions and Abbreviations F-1
F.2 Application Activity Model Diagrams F-4

Annex G - Application Reference Model (Informative) G-1
G.1 EXPRESS Definition of ARM Application Objects G-1
G.2 EXPRESS-G Diagrams of the ARM G-1

Annex H - AIM EXPRESS-G (Informative) H-1

Annex J - AIM EXPRESS Listing (Informative) J-1

Annex K - Application protocol implementation and usage guide (Informative) K-1
K.1 Pseudo-algorithm detailing the positioning and orientation of objects in space K-1

Annex M - Bibliography (Informative) M-1
M.1 ISO 10303 and Related Documents M-1
M.2 General Reference Documents M-1

Foreword

This document defines a STEP based application protocol for the exchange of product model data in the field of Thermal Analysis for Space applications, abbreviated to the acronym STEP-TAS.

The following annexes are normative: A, B, C, D.

The following annexes are informative: E, F, G, H, J, K, M.

This Application Protocol follows the guidelines and recommendations of [AP-Guide] “Guidelines for the Development and Approval of STEP Application Protocols”.

Concerning the current release

This is version 2, release 2.4 of the protocol. This is the final release of version 2 of the protocol, in which the ARM has been harmonized with the ARM of [STEP-NRF-AP] and a full mapping into the AIM is completed.

In this version of the protocol the main emphasis remains on the exchange of the data needed in the thermal-radiative analysis activity.

Introduction

0.1 Brief introduction on the STEP standard

STEP is the casual name for the ISO 10303 standard and stands for Standard for the Exchange of Product model data.Work on the standard started in 1984 and is still in progress with participants from all over the world. This introduction is compiled using elements from [STEP-1], [Kuiper-94] and [Fowler-95].

0.1.1 The objectives of STEP

STEP part 1 says: “ISO 10303 is a series of International Standards for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also a basis for implementing and sharing product databases and archiving.” [STEP-1].

In other words the STEP methodology has been developed to meet industry requirements for standard data specifications that support:

- long term storage and retention of product information;
- reduction and elimination of “islands of automation”;
- independence of data from the software tools which create or consume information;
- communication of product information between departments, disciplines, and enterprises.

In addition, the fact that STEP is a standard introduces additional requirements, in that the specification developed to fulfil these requirements should be stable, extensible, and publicly available.

0.1.2 Principles

The STEP methodology is based on a small number of fundamental principles:

- STEP defines an architecture for product data, providing stability and extensibility;
- STEP supports and requires traceability of data to industry needs;
- The role of STEP is the standardisation of industry application semantics;
- STEP defines the requirements for implementation of product data exchange, based on a separation of data specifications (the logical definition) from implementation forms (the physical realisation);
- STEP defines the requirements for the assessment of conformance of implementations.

0.1.3 The STEP architecture

The architecture of STEP results from the principles stated above. The complete architecture of STEP is described in the *ISO 10303 Architecture and Reference Manual*, which is currently being prepared for publication. Below the main elements of the architecture will be highlighted.

Layered approach with functional series of standard documents

The STEP standard takes a layered approach to product data modelling. The different layers are called series and these contain parts which are related by the function they perform. The series are given in Table 1.

Each series adds specific aspects to the product data model description. The higher numbered layers build further on concepts defined in the lower numbered layers. The EXPRESS language is used to formally define the data models (*schemas* in STEP parlance) of the parts in layers 4, 5, 6 and 7. Application protocols are directed for use by a specific user community, engineering discipline or branch of industry. The standard also includes a formal methodology for conformance testing of STEP-based products that claim to implement one of the application protocols.

Table 1 - The series of STEP standard documents

part numbers	*series title*	*example documents*
10 series	Description Methods	ISO 10303 Part 11 The EXPRESS Language Reference Manual
20 series	Implementation Methods	ISO 10303 Part 21 Clear Text Encoding of the Exchange Structure ISO 10303 Part22 Standard Data Access Interface
30 series	Conformance Testing Methodology and Framework	ISO 10303 Part 31 General Concepts
40 series	Integrated Generic Resources	ISO 10303 Part 41 Fundamentals of Product Description and Support ISO 10303 Part 42 Geometric and Topological Representation
100 series	Integrated Application Resources	ISO 10303 Part 105 Kinematics
500 series	Application Interpreted Constructs	ISO 10303 Part 509 Manifold Surface Shape Representation
200 series	Application Protocols	ISO 10303 AP 203 Configuration Controlled Design
300 series	Abstract Test Suites	ISO 10303 Part 303 Abstract Test Suite – Configuration Controlled Design (for AP 203)

Key elements of the STEP architecture

Figure 0–1 provides a high level summary of the key elements of the STEP architecture. The arrows in Figure 0–1 specify the *existence dependence* between the elements, i.e. an element at an arrow's tail is dependent on an element at its head.

After the figure a short description of each of the key elements is given.

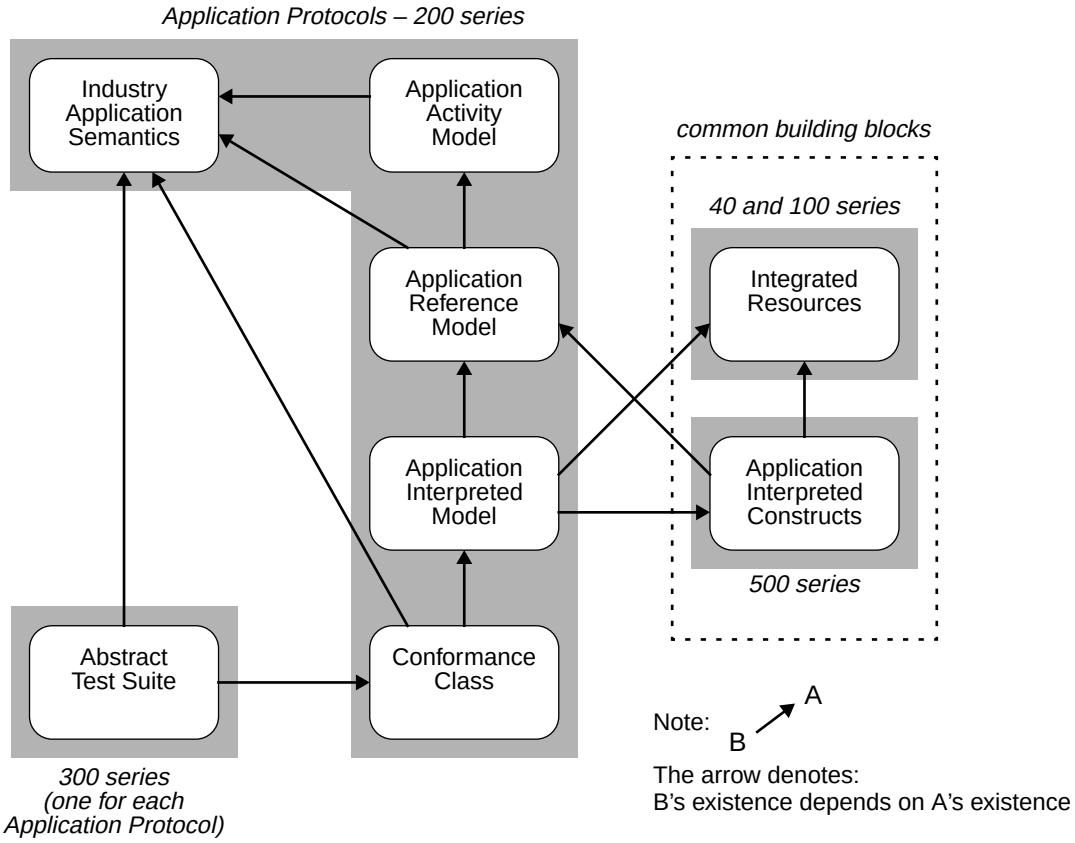


Figure 0–1 - Key elements of the STEP architecture – the grey areas denote the relation between these elements and the STEP documentation structure

Industry Application Semantics

Figure 0–1 shows that the key element upon which all other elements of the STEP architecture depend is industry application semantics. It identifies the processes and data that are essential to an industrial application domain – in other words the industrial needs and semantics that are particular to data exchange in some industry application domain. It must be possible to trace back the concepts and definitions used in the other architectural elements to the needs and semantics in the Industry Application Semantics.

Application Activity Model

Industry application semantics are defined more formally by reference to an Application Activity Model (AAM). This model is created with the IDEF0 activity modelling technique and supports the analysis of the activities and information flows within the industry application. The AAM provides a context model for the application domain and the activities and information flows that are "in scope" or "out of scope" are explicitly declared. The "in scope" information flows can then be elaborated in further detailed STEP analysis and specification models – in particular the Application Reference Model (ARM). It should be noted that the role of the AAM is to capture the activities within an industry application ("what is done"), not the detailed processes ("how it is done"). The latter are likely to vary between organisations, or with time as the result of continuous improvement or business process re-engineering.

Application Reference Model

The second element of the STEP architecture that results from detailed analysis of the product data exchange requirements in the industrial application domain, is the Application Reference Model (ARM). This is a detailed specification of the data objects (entities and attributes) and their relationships, that completely define the "in scope" information flows, which were identified in the AAM. This specification is prepared through analysis of requirements identified by experts in the

industry application – sometimes referred to as “domain experts”. These requirements are therefore described using the terminology of the application domain. They form the basis for further development as well as for review and validation, e.g. by peer experts.

Application Interpreted Model

The data exchange requirements laid down in the ARM are translated – by a “STEP expert” – into a normative STEP data specification in the Application Interpreted Model, as much as possible through selection and constraint of the common building blocks: the Integrated Resources and the Application Interpreted Constructs. This re-use of standard data constructs across a wide range of industry requirements results in a high degree of consistency and integration across models, and enables potential re-use of the software code used in interfaces and the potential sharing of common data across application domains. The AIM is defined using the EXPRESS language, which is computer interpretable and therefore enables file-based exchange according to the STEP Part 21 physical file format and/or data access using the STEP Part 22 SDAI.

Integrated Resources

The Integrated Resources are specified in context independent models and form standard data constructs that are used in the creation of an AIM. They are data models that reflect and support the common requirements of many different product data application areas. Together the Integrated Resources constitute a single, logical, conceptual product data model. They are, however, not themselves intended for direct implementation: they define re-usable building blocks that are intended to be combined and refined to meet specific requirements stated in an ARM. Integrated Resources are specified in EXPRESS.

Application Interpreted Constructs

The Application Interpreted Constructs are a relatively late addition to the STEP architecture. They capture the fulfilment of a collection of requirements that are common to two or more ARMs, i.e. shared by two or more industry application domains. Application Interpreted Constructs are also specified in EXPRESS and explicitly identify the potential for shared data or so-called *interoperability* between two or more Application Protocols / industry application domains.

The difference between Integrated Resources and Application Interpreted Constructs can explained as follows:

- Integrated Resources were developed – and are still being developed and refined – on the basis of an a priori conception by product data modelling experts of what common product data modelling resources would be needed across a wide range of application domains. As stated before Integrated Resources are not intended for direct implementation in an AIM, but often need to be combined and refined first.
- Application Interpreted Constructs are the common collections of requirements in the ARM and their corresponding implementation in the AIM that are “discovered” during the development of various Application Protocols. Application Interpreted Constructs are a ready-for-implementation kind of “plug-in” modules that can be included in an AIM.

Conformance Classes

An Application Interpreted Model gives the normative specification for data to be exchanged between computer applications. This provides the scope and boundaries for implementations of product data exchange that conform to STEP, and also the cope and boundaries for testing implementations. In order to meet the needs of differing computer systems used within a given industrial application, whilst maintaining consistency of implementation and testing, two or more Conformance Classes may be specified for an AIM. A Conformance Class defines a subset of the AIM that may be used as the basis for implementation and testing. These subsets define the minimum conforming implementation based on the AIM; implementations based on any other subsets are not considered to be conforming.

Abstract Test Suites

The importance of testing and testability within STEP is reflected by a standardised framework and methodology for conformance testing. The Abstract Test Suite is the manifestation of the needs of testing within the STEP architecture. An Abstract Test Suite specifies, in non-specific or parameterised form, the test cases that will be used in assessing the conformance of an implementation to the data specification contained in an AIM and thee other elements of the STEP architecture upon which an

AIM depends. Experience in other domains, such as the OSI standard for Open Systems, has shown that standardisation of Abstract Test Suites is an essential prerequisite to repeatability and consistency of testing, and therefore of mutual recognition of test results across regional or national bodies.

0.1.4 The basis of the STEP product data model

The consistency of data specifications within STEP for a wide range of industry applications – Application Protocols – is ensured by the reuse of common Integrated Resources. The Integrated Resources themselves are based in a formalised framework for product data, sometimes referred to as the Generic Product Data Model. The framework defines the basis of all data specification that are standardised within STEP.

Elements of data specifications (or “constructs”) are taken to be the representation of facts about objects in the real world. The Generic Product Data Model framework is based on a classification of the types of data that describe products. The classification identifies five major types of data:

- *The application context* – data that defines the purpose for which product information is created, and the types of product, disciplines, and life-cycle stages for which such information is valid. The use of an application context allows e.g. data that represents an “as designed” product to be distinguished from that for an “as built” configuration, etc.
- *The definition of a product* – data that identifies products, including variants, categories and/or life-cycle “views” of products. Product definition data also includes that which relates to the structure of products, in terms of assembly structures, configurations, effectivities, bill of material, etc.
- *The definition of product properties* – data that characterises products by their properties, independent of the representation of such properties. For example, it is possible to identify the shape of an object, or aspects of the shape, as a property of the object, without providing a detailed description of the shape using a CAD model or an engineering drawing. Examples of product properties are: shape, surface finish, constituting material.
- *The representation of product properties* – data that represents the properties of a product, including multiple representations of the same property. For example, the shape of an object may be identified, and then represented in many different ways: a 3D CAD CSG model, a physical mock-up, an engineering drawing, a technical illustration and a mathematical model are different representations of the same shape.
- *The presentation of the representation of product properties* – data that defines the presentation of product information to support human communication. For example, the shape of an object (which is a product property) is represented by coordinate values, curves, surfaces, etc.; then this representation is presented by assigning colours, line fonts and a viewpoint, and displaying the resulting picture on a workstation.

This classification of product data is the basis for all STEP data specifications. It is the framework upon which all the Integrated Resources are built, and is reflected in the Application Interpreted Models (AIMs) of all Application Protocols. The models that capture this framework embody the principle of existence dependence, which ensures that all product information is related to an identified product and ultimately to an application context. This classification structure is summarised graphically in Figure 0–2.

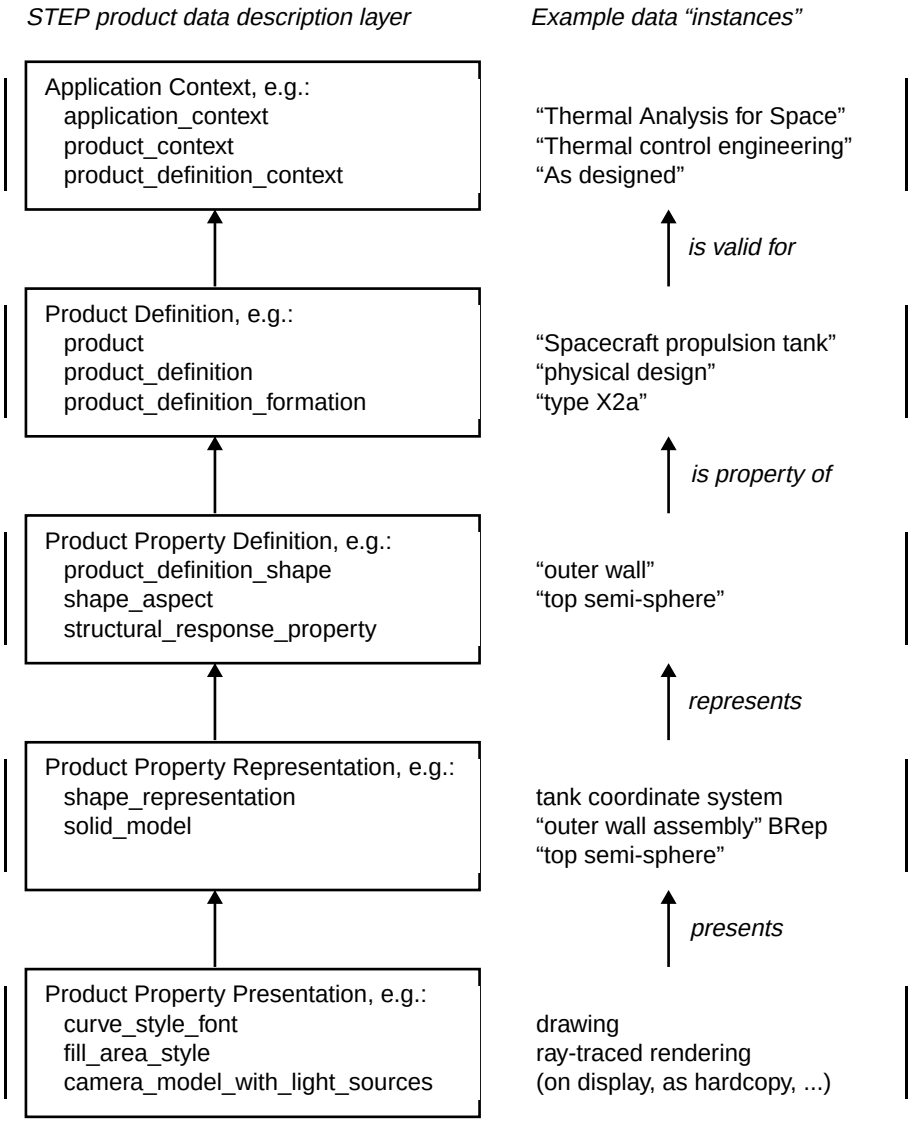


Figure 0–2 - Product data type classification and existence dependence in STEP data models

This principle can lead to models that are at first sight counter-intuitive: rather than stating that a product has a shape, a shape is "of" a product. However, simple analysis of this example shows that the existence dependent form of the model requires that a shape is always the shape of a product. Similarly, at a lower level in the structure, STEP does not allow the existence of "geometry" data just as collections of points, lines, curves, etc. Through the existence dependent structures in the STEP models, such a collection of geometry data must be the representation of some property, that is related to the definition of a product, that has validity in some application context. Thus the basic structure of the STEP models satisfies and enforces the principle identified above: that all product data should be traceable to an industrial need.

0.1.5 Realisation of a STEP solution for product data exchange

One of the main features of the STEP standard is the division between *logical* and *physical* representation of model data.This helps in achieving a clean development process which is best illustrated by the following diagram.

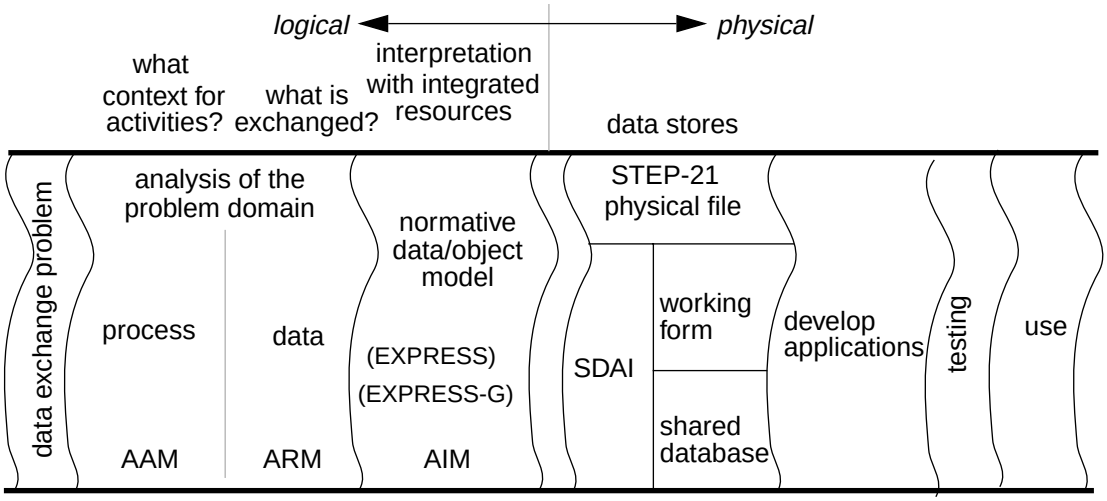


Figure 0–3 - Illustration of the development steps needed to reach a working STEP product data exchange solution and different elements of the STEP standardisation methodology (adapted from a diagram by P. Kuiper – TNO)

Actual data exchange takes place through the physical implementation in either a [STEP-21] physical file, a so-called working form or shared database. The physical implementation is derived in a standardised way from the logical definition in an Application Interpreted Model (AIM) of a certain Application Protocol and can possibly be accessed through an SDAI [STEP-22] compliant programming interface, for which a significant part could have been computer-generated from the AIM EXPRESS schema by a suitable STEP toolkit.

0.1.6 Status of the STEP standard and its implementations

The STEP standard is becoming mature and the First Release (Parts 1, 11, 21, 31, 41, 42, 43, 44, 46, 101, 201, 203) has been issued per September 1994 as International Standard (IS). Commercial development tools are available on the market and most major CAx software vendors are now offering STEP-based import/export facilities or have announced to support the STEP standard.

0.2 The Industrial Need for STEP -based Data Exchange for Thermal Analysis for Space

The thermal control engineering of spacecraft and other space-related equipment requires extensive computer-supported analyses and tests, at different levels of detail. In typical space projects several companies and/or institutions work together. They are often located at different sites, which may be geographically far apart and they often use different computer platforms. Efficient and reliable electronic exchange of model and results data is therefore highly needed.

Currently, the exchange of such data is done via a great number of dedicated – mostly tool-specific – data formats and the use of many point-to-point converter tools. These tools convert data in one format as well as they can into data in another format. The proliferation of data formats and corresponding converter tools place a great burden on software development and maintenance resources. The converter tools need to be regularly updated as data formats change or get extended at either side of an interface. There is often a loss of information in the conversion process, because finding and implementing the best mapping of concepts between two tool-specific data formats is a non-trivial task and normally a converter developer controls only one of the two data format interfaces.

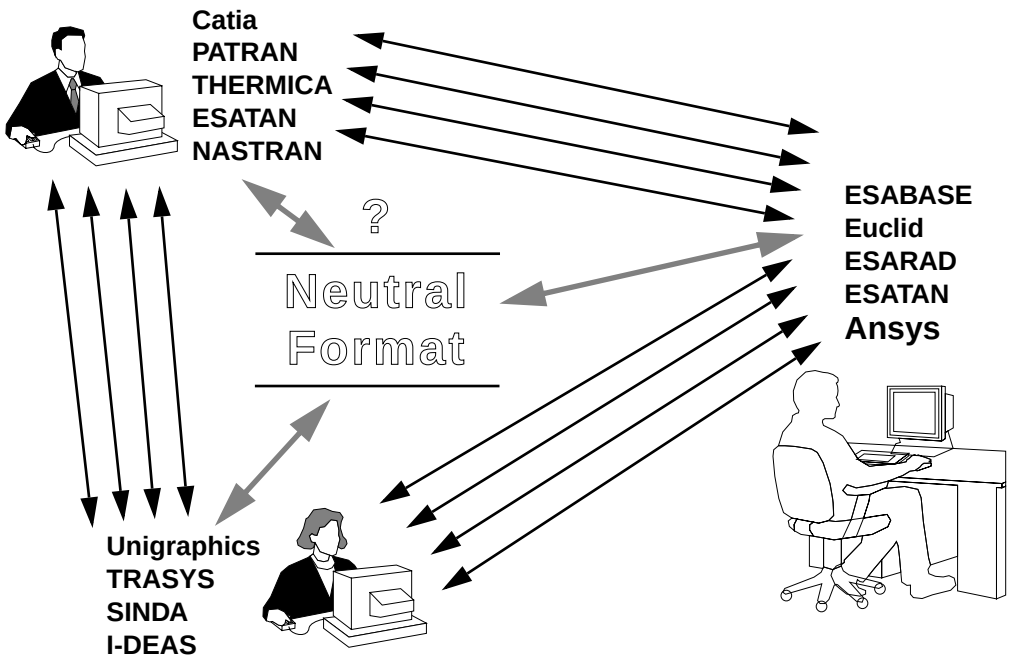


Figure 0–4 - Illustration of the problem of model and data exchange in thermal engineering for space (with an arbitrary selection of tools)

Apart from direct data exchange between e.g. a main contractor and a client or a sub-contractor and its main contractor, there is also a need for a stable neutral format for long term electronic archiving and a growing need to support concurrent engineering: i.e. exchanging models and results with other engineering disciplines during the design and development process, e.g. for thermo-elastic analysis. In the long term, ideally concurrent engineering would be supported by having shared databases which are simultaneously accessible to all engineers of different disciplines in a project team.

The current STEP-TAS Application Protocol attempts to lay the foundation to address these needs.

An important additional possibility of this STEP based protocol is that it is defined in such a generic way that it could be extended in the future to cover not only space thermal analysis, but also the related field of space environmental analysis disciplines.

Clause 1 - Scope

The scope of the STEP-TAS Application Protocol comprises the data exchanged in the following activities in the domain of thermal control engineering for space applications:

1) Define & Formulate Thermal Control Design, which is subdivided into:
   a) Formulate Thermal Control Design
      Input data: physical properties of standard parts and materials, product structure, CAD definition, analysis & test results
      Output data: product description
   b) Define Thermal Control Analysis & Test Cases
      Input data: product description, analysis & test models
      Output data: analysis and test cases
   c) Construct Thermal Control Analysis & Test Models
      Input data: product description, analysis & test cases, product structure, CAD definition
      Output data: analysis and test models
2) Analyse & Test Thermal Control Design, which is subdivided into:
   a) Do Thermal-Radiative Analysis
      Input data: analysis cases, analysis models
      Output data: analysis results – e.g. radiative exchange factors, environmental fluxes
   b) Do Thermal Network Analysis
      Input data: analysis cases, analysis models, analysis results from thermal-radiative analysis
      Output data: analysis results – e.g. temperature fields, heat balances
   c) Correlate Analysis and Test Results
      Input data: analysis results (predictions), test results (measured)
      Output data: analysis results (correlations)

A complete definition of the activities and data flows relevant to thermal control engineering for space is given in:

- Annex F - “Application Activity Model (Informative)” on page F-1.

The requirements for the data to be exchanged are defined in detail in:

- Clause 4 - “Information Requirements” on page 4-1 and
- Annex G - “Application Reference Model (Informative)” on page G-1.

The actual definition of the data that can be exchanged with this Application Protocol can be found in:

- Clause 5 - “Application Interpreted Model” on page 5-1 and
- Annex A - “AIM EXPRESS expanded listing (normative)” on page A-1.

Note for Version 2

Activities 1a, 1b, 1c and 2a are currently in scope – the main focus is on thermal-radiative modelling; activities 2b and 2c are now covered by the [STEP-NRF-AP].

Clause 2 - Normative References

The following documents, ISO standards or working drafts contain provisions which, through reference in this text, constitute provisions of this Application Protocol. At the time of development, the editions indicated between parenthesis were valid.

[STEP-1]	ISO 10303-1:1994 (IS) Overview and fundamental principles
[STEP-11]	ISO 10303-11:1994 (IS) Description methods: The EXPRESS language reference manual
[STEP-21]	ISO 10303-21:1994 (IS) Implementation methods: Clear text encoding of the exchange structure Note: Also known as the “STEP Physical File” format.
[STEP-31]	ISO 10303-31:1994 (IS) Conformance testing methodology and framework
[STEP-41]	ISO 10303-41:1994 (IS) Integrated generic resources: Fundamentals of product description and support
[STEP-42]	ISO 10303-42:1994 (IS) Integrated generic resources: Geometric and topological representation
[STEP-43]	ISO 10303-43:1994 (IS) Integrated generic resources: Representation structures
[STEP-44]	ISO -10303-44:1994 (IS) Integrated generic resources: Product structure configuration
[STEP-45]	ISO TC184/SC4/WG3 N354 (FDIS) Integrated generic resources: Materials
[STEP-46]	ISO 10303-46:1994 (IS) Integrated generic resources: Visual presentation
[STEP-105]	ISO/DIS 10303-105 (IS) Integrated application resources: Kinematics
[STEP-509]	ISO 10303-509 (CD) Application interpreted constructs: Manifold surface
[STEP-511]	ISO 10303-511 (CD) Application interpreted constructs: Topology bounded surface
[PLIB-20]	ISO 13584-20 (DIS) Parts library: General resources
[PLIB-42]	ISO 13584-42 (DIS) Parts library: Methodology for structuring part families

Note: the edition (release status) of the documents is given between parentheses, using the following abbreviations:

WD	Working Draft
CD	Committee Draft
CDB	Committee Draft, in Ballot
CDBE	Committee Draft, Ballot Ended
CDC	Committee Draft, for Comment
CDCE	Committee Draft, Comments Ended

DIS	Draft International Standard
IS	International Standard

Clause 3 - Definitions and Abbreviations

3.1 Terms Defined in ISO 10303-1

3.1.1 **abstract test suite** – a part of this International Standard that contains the set of abstract test cases necessary for conformance testing of an implementation of an application protocol.

3.1.2 **application** – a group of one or more processes creating or using product data.

3.1.3 **application activity model (AAM)** – a model that describes an application in terms of its processes and information flows.

3.1.4 **application context** – the environment in which the integrated resources are interpreted to support the use of product data in a specific application.

3.1.5 **application interpreted model (AIM)** – an information model that uses the integrated resources necessary to satisfy the information requirements and constraints of an application reference model, within an application protocol.

3.1.6 **application object** – an atomic element of an application reference model that defines a unique concept and contains attributes specifying the data elements of the object.

3.1.7 **application protocol** – ~~a part of this International Standard~~ *a document complying with [AP-Guide]* that specifies an application interpreted model satisfying the scope and information requirements for a specific application.

3.1.8 **application reference model (ARM)** – an information model that describes the information requirements and constraints of a specific application.

3.1.9 **application resource** – an integrated resource whose contents are related to a group of application contexts.

3.1.10 **assembly** – a product that is decomposable into a set of components or other assemblies from the perspective of a specific application.

3.1.11 **component** – a product that is not subject to decomposition from the perspective of a specific application.

3.1.12 **conformance class** – a subset of an application protocol for which conformance may be claimed.

3.1.13 **conformance requirement** – a precise, text definition of a characteristic required to be present in a conforming implementation.

3.1.14 **data** – a representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers.

3.1.15 **data exchange** – the storing, accessing, transferring, and archiving of data.

3.1.16 **data specification language** – a set of rules for defining data and their relationships suitable for communication, interpretation, or processing by computers.

3.1.17 **exchange structure** – a computer-interpretable format used for storing, accessing, transferring, and archiving data.

3.1.18 **generic resource** – an integrated resource whose contents are context-independent.

3.1.19 **implementation method** – a part of this International Standard that specifies a technique used by computer systems to exchange product data that is described using the EXPRESS data specification language [STEP-11].

3.1.20 **information** – facts, concepts, or instructions.

3.1.21 **information model** – a formal model of a bounded set of facts, concepts or instructions to meet a specified requirement.

3.1.22 **integrated resource** – a part of this International Standard (i.e. ISO 10303) that defines a group of resource constructs used as the basis for product data.

3.1.23 **interpretation** – the process of adapting a resource contruct from the integrated resources to satisfy a requirement of an application protocol. This may involve the addition of restrictions on attributes, the addition of constraints, the addition of relationships among resource constructs and application constructs, or all of the above.

3.1.24 **PICS proforma** – a standardised document in the form of a questionnaire, which, when completed for a particular implementation, becomes the protocol implementation conformance statement.

3.1.25 **presentation** – a recognizable visual representation of product data.

3.1.26 **product** – a thing or substance produced by a natural or artificial process.

3.1.27 **product data** – a representation of information about a product in a formal manner suitable for communication, interpretation, or processing by human beings or by computers.

3.1.28 **product information** – facts, concepts, or instructions about a product.

3.1.29 **product information model** – an information model which provides an abstract description of facts, concepts and instructions about a product.

3.1.30 **protocol implementation conformance statement (PICS)** – a statement of which capabilities and options are supported within an implementation of a given standard. This statement is produced by completing a PICS proforma.

3.1.31 **resource construct** – a collection of EXPRESS language entities, types, functions, rules and references that together define a valid description of an aspect of product data.
Refer to ISO 10303 part 11 for the reference manual of the EXPRESS language.

3.1.32 **structure** – a set of interrelated parts of any complex thing, and the relationships between them.

3.1.33 **unit of functionality** – a collection of application objects and their relationships that defines one of more concepts within the application context such that removal of any component would render the concept incomplete or ambiguous.

3.2 Terms Defined in ISO 10303-31

3.2.1 **postprocessor** – a software unit that translates product information from an independent public domain product data format to the internal format of a particular computer system.

3.2.2 **preprocessor** – a software unit that translates product information from the internal format of a particular computer system to an independent public domain product data format.

3.3 Terms Defined in ISO 10303-42

3.3.1 **geometric coordinate system** – the underlying global rectangular Cartesian coordinate system to which all geometry refers.

3.3.2 **surface** – a set of mathematical points which is the image of a continuous function defined over a connected subset of the plane (R^2). This image shall not be a single point or in part, or entirely, a curve.

3.4 Terms Defined in ISO 10303-45

3.4.1 **material** – the substance or substances of which product is composed or made.

3.4.2 **material property** – a product characteristic which depends upon the material or materials comprising the product.

3.4.3 **material designation** – an identifier which is assigned by agreement.

3.5 Terms Defined in ISO 10303-105

NOTE 1 – For the definitions of terms “base”, “ground”, “link”, “joint” and “pair” the adjective “kinematic” has been added.

3.5.1 **frame** – a coordinate system used in kinematic representations.

3.5.2 **kinematic base** – a kinematic link which is fixed to the kinematic ground or whose motion is prescribed along a defined path.

3.5.3 **kinematic chain** – a sequence of kinematic links connected by kinematic joints.

3.5.4 **kinematic ground** – the immobile part of a mechanical product in the ~~world~~ basic coordinate system

3.5.5 **kinematic joint** – the topological aspect of an ordered connection or motion restriction between two and only two kinematic links.

3.5.6 **kinematic link** – a rigid kinematic object whose motion is constrained by one or more kinematic joints.

NOTE 2 – In STEP-TAS **kinematic link** is a synonym for **rigid body**.

3.5.7 **kinematic pair** – the geometric aspect of either an ordered connection or motion restriction between two and only two kinematic links.

3.5.8 **placement** – the location and orientation of a frame.

3.6 Terms Defined in this AP

NOTE 3 – These definitions are partly based on [ICETAS-URD], [ATS-URD-v2], [ESARAD-RM], [THERMICA], [ESATAN-RM], [ESABASE-RM], [ESABASE-THM], [Larson & Wertz].

3.6.1 **altitude** – distance from the surface of a celestial body to the origin of the coordinate system of the main body of a space product in an orbit around a celestial body.

3.6.2 **apoapsis** – orbit position farthest from the gravitational body that governs the orbit.

3.6.3 **attitude** – orientation of a rigid body that is (part of) a space product.

3.6.4 **celestial body** – a natural body in space, e.g. a star, a planet, a moon, a comet, an asteroid.

3.6.5 **enclosure** – set of thermal-radiative faces which can exchange thermal radiation. The enclosure concept is used to increase the efficiency of thermal radiation computations by subdividing the thermal-radiative model into sets of thermal-radiative faces which form enclosed spaces. Thermal-radiative faces in such an enclosure can only exchange thermal radiation with each other and not with thermal-radiative faces outside the enclosure.

3.6.6 **exchange dataset**– coherent and valid set of instances of entities conforming to the AIM schema defined in this AP. The dataset is instantiated in the form of a [STEP-21] physical file, a working form or a shared database. An exchange dataset can be accessed through an SDAI programming interface, as defined in [STEP-22].

3.6.7 **gauge** – a means or device for measurement.

3.6.8 **mission** – the collection of primary and derived goals to be fulfilled by a space product. Aspects of a mission comprise its orbit trajectories, attitudes, space environments, operational modes and sequence of events. A mission may be subdivided into mission phases.

3.6.9 **model** – a representation in software or hardware of one or more properties of a (possibly conceptual) product and its (intended) operating environment for the purpose of design, analysis or verification. By its nature a model is an idealised representation of a product.

3.6.10 **multiple reflection** – thermal radiation travelling from one thermal-radiative face to another thermal-radiative face via at least one intermediate reflection.

3.6.11 **orbit** – trajectory in space, governed by gravitational forces and/or propulsion.

3.6.12 **orbit arc** – part of an orbit.

3.6.13 **periapsis** – orbit position nearest to the celestial body that governs the orbit.

3.6.14 **radiative coupling** – the radiative coupling between two thermal-radiative faces i and j is the product of the surface area of face i and the radiative exchange factor from i to j and the emissivity of face i, or – alternatively – the product of the surface area of face j and the radiative exchange factor from j to i and the emissivity of face j.

3.6.15 **radiative exchange factor** – the radiative exchange factor between two thermal-radiative faces i and j is the fraction of the thermal radiation emitted by i that is finally – possibly after multiple reflections – absorbed by j.

3.6.16 **rigid body** – a rigid kinematic object, i.e. a collection of parts of a product which can not move one relative to another. The collection of parts is considered to be rigid at least for the purpose of thermal analysis. As a consequence movement is only possible at interfaces between rigid bodies. The boundaries of a rigid body coincide with the boundaries of a complete submodel.

3.6.17 **root-model** – The highest level model in a hierarchical model/submodel tree. The root-model comprises directly or indirectly all submodels.

3.6.18 **run** – an elementary analysis, test or operation step with a clearly defined start and end, performed in the framework of an analysis study, a test or an operation. In a run an analysis model, a test article or an operating product is analysed, tested or operated for a particular purpose and in a particular environment, using a particular analysis tool, a test facility or (when applicable) an operational facility. A run produces analysis, test or operation results.

3.6.19 **submodel** – a next lower level model in a hierarchical model/submodel tree. A submodel may contain other (yet lower level) submodels. The model/submodel tree forms an acyclic graph.

3.6.20 **submodel occurrence** – a reference to a model at a level lower than root-model level in a hierarchical model/submodel tree. The submodel occurrence has some attributes in addition to the model it references: e.g. its usage name and – for geometric models – its location and orientation with respect to its supermodel.

3.6.21 **supermodel** – a next higher level model in a hierarchical model/submodel tree. A supermodel contains one or more submodels. The model/submodel tree forms an acyclic graph.

3.6.22 **thermal lumped parameter model** – thermal mathematical model using the lumped parameter nodal approach, where the physical properties of the lumped parameter nodes are considered to be uniform – i.e. a node is iso-thermal and has uniform thermo-optical and thermo-physical properties. The model comprises a nodal breakdown and a network of links between the nodes. Such a model is also sometimes called a Thermal Network Model, because of the analogy with electrical networks. The purpose of the model is to represent a product for the purpose of thermal analysis computations, in particular of temperatures, heat flows and power.

3.6.23 **thermal node** – lumped parameter node, which is the discretisation atom in a thermal lumped parameter model.

3.6.24 **thermal-radiative face** – part of a bounded (i.e. finite) surface, with orientation, used as a discretisation atom in a thermal-radiative model. These faces possess thermo-optical properties and represent boundary surfaces at which thermal radiative heat transfer can take place.

3.6.25 **thermal-radiative face set** – set of thermal-radiative faces which are associated with one side of one bounded surface and share identical thermo-optical properties.

3.6.26 **thermal-radiative model** – geometrical surface model representing a product for the purpose of thermal radiation analysis, in particular for computation of radiative exchange factors and environmental thermal fluxes. The surfaces in the model are associated with thermal-radiative face sets.

3.6.27 **viewfactor** – also called geometric viewfactor – fraction which gives a measure for the direct view from one face to another face. The viewfactor from face i to face j is defined as the quotient of the solid angle representing the direct view from i to j over the solid angle of total possible view from face i. For a flat face the total possible view is a solid angle of 2π sterradians, i.e. a hemisphere.

3.7 Abbreviations

AAM	Application Activity Model
AIM	Application Interpreted Model
AP	Application Protocol
ARM	Application Reference Model
B-rep	Boundary Representation Solid Model
DBMS	Data Base Management System
CNES	Centre National d'Etudes Spatiales – French Space Agency
CSG	Constructive Solid Geometry
ESA	European Space Agency
ESATAN	ESA's Thermal Network Solver
ESARAD	ESA's thermal radiative computation software
ESTEC	ESA - European Space Technology Centre
FDIR	Failure Detection, Isolation and Recovery
FLUOR	CNES' *Flux en Orbit* thermal radiation computation software
FMECA	Failure Mode Effects and Criticality Analysis
GEO	Geostationary Earth Orbit
ICETAS	Integrated Communication Environment for Thermal Analysis Software
ISO	International Organization for Standardization
LEO	Low Earth Orbit
NMBR	Network Model Bulk Results (protocol, developed in the Network Results Format project)
OO	Object Oriented
RAMS	Reliability, Accessibility, Maintainability and Safety
SDAI	Standard Data Access Interface (defined in [STEP-22])
SET	*Standard d'Echange et de Transfert* (French Standard AFNOR Z68-300)
SET-ATS	SET Protocol d'Application Thermique Spatiale
SI	*Système International des Unités* - International System of Units
STEP	Standard for the Exchange of Product Model Data (ISO 10303)
STEP-TAS	STEP-based Thermal Analysis for Space
TBD	To be defined
URD	User Requirements Document
UoF	Unit of Functionality

Clause 4 - Information Requirements

This clause specifies in detail what kind of data needs to be exchanged in the domain of thermal analysis for space. The information requirements are grouped in three subclauses: Units of Functionality, Application Objects and Application Assertions.

A graphical representation of the information requirements is given in Annex G "Application Reference Model (Informative)" on page G-1.

The information requirements correspond to the data streams which are identified as being in scope for this Application Protocol in Annex F "Application Activity Model (Informative)" on page F-1.

4.1 Units of Functionality

This subclause specifies the units of functionality for the STEP AP - Thermal Analysis for Space.

The first seven UoFs are identical to those of the [STEP-NRF-AP]:

1) **product_structure** UoF
2) **network_model_representation** UoF
3) **bulk_results** UoF
4) **parameterized_functions** UoF
5) **physical_quantities_and_units** UoF
6) **date_and_time** UoF
7) **general_support** UoF

The UoFs following hereafter are particular to the STEP-TAS protocol:

8) **product_requirements** UoF
9) **space_mission_aspects** UoF
10) **geometric_model** UoF
11) **kinematic_model** UoF
12) **thermal_radiative_model** UoF
13) **thermal_lumped_parameter_model** UoF
14) **thermal_finite_element_model** UoF
15) **visual_representation** UoF
16) **material_properties** UoF
17) **additional_physical_quantities_and_measures** UoF

Hereafter each of the Units of Functionality is described in more detail.

4.1.1 product_structure UoF

The **product_structure** UoF captures the structure of the product being modelled – e.g. the product assembly tree or a functional decomposition tree. The definition of this UoF already follows closely the product structure definition in [STEP-41] and [STEP-44].

NOTE 4 – This Unit of Functionality is defined following as closely as possible the concepts for product and product structure definition in [STEP-41] and [STEP-44].

The following application objects are used by the **product_structure** UoF:

- **product**
- **product_context**
- **product_definition**
- **product_definition_context**
- **product_next_assembly_usage_relationship**
- **product_version**

4.1.2 network_model_representation UoF

The **network_model_representation** UoF collects the definitions needed for a generic representation of engineering objects as a hierarchical structure of models containing a network of discrete nodes and relationships between these nodes.

The following application objects are used by the **network_model_representation** UoF:

- **model_represents_product_relationship**
- **network_model**
- **network_model_component**
- **network_model_constituent**
- **network_node**
- **network_node_class**
- **network_node_relationship**
- **network_node_relationship_class**
- **network_node_relationship_usage**
- **network_node_usage**
- **submodel_usage**

4.1.3 bulk_results UoF

The **bulk_results** UoF collects the application objects needed for the identification and definition of analysis, test or operation campaigns, cases, phases and runs and the results data they produce. The results data are characteristic, predicted, assigned, observed and/or derived properties of network models, of which the values may need to be represented as scalars, vectors or higher order tensors.

The following application objects are used by the **bulk_results** UoF:

- **abscissa_sequencing_type**
- **ato_aspect**
- **ato_campaign**
- **ato_case**
- **ato_phase**
- **lifetime_type**
- **list_of_descriptive_values**
- **list_of_functions**
- **model_component_sequence**
- **property_class**
- **property_class_scalar**
- **property_class_scalar_descriptive**
- **property_class_scalar_functional**
- **property_class_scalar_quantitative**
- **property_class_tensor**
- **property_class_usage**
- **property_name**
- **property_role**
- **property_symmetry**
- **run**
- **run_sequence**
- **scalar_in_tensor**
- **scalar_value**
- **scan**
- **scan_derivation_procedure**
- **scan_mask**

- **scan_of_derived_values**
- **scan_of_sampled_values**
- **unit_assignment**

4.1.4 parameterized_functions UoF

The **parameterized_functions** UoF contains the application objects needed to store functions of one or more independent parameters, which are physical quantities. Functions which can be expressed in tabular or polynomial form or as textual formulae are considered.

The following application objects are used by the **parameterized_functions** UoF:

- **cyclic_tabular_function**
- **parameterized_function**
- **parameterized_function_with_formula**
- **polynomial_function**
- **tabular_function**
- **tabular_interpolation_type**

4.1.5 physical_quantities_and_units UoF

The **physical_quantities_and_units** UoF captures the definition and representation of physical quantities, their dimensions, units and values.

The following application objects are used by the **physical_quantities_and_units** UoF:

- All objects from the **measure_schema** defined in clause 4.14 of [STEP-41].

In particular:

- **unit**

4.1.6 date_and_time UoF

The **date_and_time** UoF allows representation combinations of calendar date and time of day to be defined.

NOTE 5 – The objects defined in this UoF correspond one-to-one with those defined in the **date_time_schema** of [STEP-41]. Here only copies of the directly used objects are given.

NOTE 6 – The [STEP-41] **ordinal_date**, **day_of_week** and **week_of_year** objects are out of scope.

The following application objects are used by the **date_and_time** UoF:

- **calendar_date**
- **coordinated_universal_time_offset**
- **date**
- **date_and_time**
- **local_time**

4.1.7 general_support UoF

The **general_support** UoF contains application objects that are used to support the use of the principal application objects listed in the other Units of Functionality in this subclause.

NOTE 7 – The objects in this UoF are taken as much as possible literally from [STEP-41] and [STEP-203].

The following application objects are used by the **general_support** UoF:

- **address**
- **approval**
- **document**
- **document_type**
- **document_usage_constraint**
- **organization**
- **organizational_address**
- **organizational_project**

- **person**
- **person_and_organization**
- **personal_address**
- **security_classification_level**

4.1.8 space_mission_aspects UoF

The **space_mission_aspects** UoF is a structured collection of data on the mission to be fulfilled by the space product being developed or operated.

The following application objects are used by the **space_mission_aspects** UoF:

- **space_mission**
- **space_mission_case**
- **space_mission_phase**
- **space_mission_aspect**
- **space_mission_event**
- **space_mission_event_with_epoch**
- **space_mission_event_with_running_time**
- **space_mission_event_with_true_anomaly**
- **space_coordinate_system**
- **orbit_arc**
- **discretised_orbit_arc**
- **keplerian_orbit_arc**
- **kepler_parameter_set**
- **orbit_position_and_velocity**
- **celestial_body**
- **celestial_body_with_orbit**
- **thermal_aspects_of_celestial_body**
- **albedo_reflector**
- **main_body_placement**
- **main_body_placement_with_constant_rotation**
- **main_body_placement_with_spinning**
- **pointing_in_space**
- **general_star_pointing**
- **general_space_pointing**
- **attitude_rotation**
- **thermal_environment**
- **thermal_radiation_spectrum**
- **operational_mode**
- **degradation_dose_profile**
- **kinematic_articulation**

4.1.9 geometric_model UoF

The following application objects are used by the **geometric_model** UoF:

- **geometric_context**
- **geometric_model**
- **geometric_submodel_usage**
- **geometric_construction_point**
- **axis_placement**
- **primitive_bounded_surface**
- **triangle**
- **rectangle**

- **quadrilateral**
- **disc**
- **cylinder**
- **cone**
- **sphere**
- **paraboloid**
- **compound_surface_shell**
- **boolean_construction_surface**
- **half_space_plane**
- **half_space_cylinder**
- **generalized_half_space**
- **boolean_result_surface**
- **boolean_difference_surface**

4.1.10 kinematic_model UoF

The following application objects are used by the **kinematic_model** UoF:

- **kinematic_model**
- **kinematic_joint**
- **kinematic_pair**
- **kinematic_pair_value**
- **revolute_kinematic_pair**
- **explicit_revolute_kinematic_pair_value**
- **implicit_revolute_kinematic_motion**
- **prismatic_kinematic_pair**
- **prismatic_kinematic_pair_value**

4.1.11 thermal_radiative_model UoF

The following application objects are used by the **thermal_radiative_model** UoF:

- **basic_thermal_radiative_model**
- **thermal_radiative_face_set**
- **meshed_thermal_radiative_model**
- **thermal_radiative_meshing**
- **thermal_radiative_meshing_with_incremental_node_list**
- **thermal_radiative_meshing_with_full_node_list**
- **enclosure**

4.1.12 thermal_lumped_parameter_model UoF

WORKING NOTE 1 – This item is identified for a future version and not in scope for the AIM.

The **thermal_lumped_parameter_model** is a type of **network_model** representing the thermal lumped parameter network model, which is used to compute temperatures and heatflows.

4.1.13 thermal_finite_element_model UoF

WORKING NOTE 2 – This item is identified for a future version and not in scope for the AIM.

4.1.14 visual_presentation UoF

The **visual_presentation** UoF collects the application objects that capture visualisation aspects for use in the various model representations.

The following application objects are used by the **visual_presentation** UoF:

- **colour_rgb**

4.1.15 material_properties UoF

The **material_properties** UoF contains the application objects needed to specify identified materials – possibly with surface treatments – and store their physical properties, as they are valid in certain environments.

The following application objects are used by the **material_properties** UoF:

- **material**
- **product_material_relationship**
- **material_property**
- **material_property_environment**
- **commercial_aspects_of_material**

4.1.16 additional_physical_quantities_and_measures UoF

The **additional_physical_quantities_and_measures** UoF contains the application objects concerning physical quantities and measures which are needed in addition to those specified in [STEP-NRF-AP].

The following application objects are used by the **additional_physical_quantities_and_measures** UoF:

- **derived_unit**
- **conversion_unit**
- **function_valued_property**
- **string_valued_property**
- **number_valued_property**

4.2 Application Objects

This subclause specifies the application objects for the STEP-TAS application protocol. Each application object is an atomic element that embodies a unique application concept and contains attributes specifying the data elements of the object.

The specification of each object comprises the following:

- its definition and purpose;
- its attributes, including its associations to other application objects;
- the assertions that pertain directly to the object itself.

The application objects are grouped by Unit of Functionality.

Only the application objects that are specific to STEP-TAS are defined here. For the definition of the application objects re-used from the STEP-NRF UoFs, the reader is referred to subclause 4.2 of [STEP-NRF-AP].

Application assertions constraining [STEP-NRF-AP] application objects included in STEP-TAS, are also valid within the present application protocol. When applicable, additional constraints on these application objects are added in the clause 4.3 of this protocol.

4.2.1 product_structure UoF

The application objects pertaining to this UoF are defined in subclause 4.2 of [STEP-NRF-AP].

4.2.2 network_model_representation UoF

The application objects pertaining to this UoF are defined in subclause 4.2 of [STEP-NRF-AP].

4.2.3 bulk_results UoF

The application objects pertaining to this UoF are defined in subclause 4.2 of [STEP-NRF-AP].

4.2.4 parameterized_functions UoF

The application objects pertaining to this UoF are defined in subclause 4.2 of [STEP-NRF-AP].

4.2.5 physical_quantities_and_units UoF

The application objects pertaining to this UoF are defined in subclause 4.2 of [STEP-NRF-AP].

4.2.6 date_and_time UoF

The application objects pertaining to this UoF are defined in subclause 4.2 of [STEP-NRF-AP].

4.2.7 general_support UoF

The application objects pertaining to this UoF are defined in subclause 4.2 of [STEP-NRF-AP].

4.2.8 space_mission_aspects UoF

4.2.8.1 space_mission

A **space_mission** is a type of **ato_campaign** which specifies a mission in space. It collects the overall characteristics and assigned data that apply to a number of **space_mission_case**s.

Attributes:

- None

Assertions:

- None

Notes:

NOTE 8 – The attributes and assertions are in addition to those inherited from **ato_campaign**.

4.2.8.2 space_mission_case

A **space_mission_case** is a type of **ato_case** which specifies a sequence of **space_mission_phase**s. The ordering in sequence and synchronisation of these **space_mission_phase**s is achieved through the use of **space_mission_event**s.

Attributes:

- *coordinate_system*: reference to a **space_coordinate_system**

Assertions:

- The *of_campaign* shall be a **space_mission**
- All *phases* shall be **space_mission_phase**s
- The *root_model* shall reference a **basic_thermal_radiative_model** or a **meshed_thermal_radiative_model**

Notes:

NOTE 9 – The attributes and assertions are in addition to those inherited from **ato_case**.

Examples:

EXAMPLE 1 – An example **space_mission_case** could be one with the following attributes:
id is "LEO02v1.1"
name is "LEO Hot Earthpointing"
description is "Nominal lower altitude sun-synchronous orbit with morning passage – Satellite earthpointing with yaw steering – Thermal hot case conditions"
phases reference a number of **space_mission_phase**s for:

- a **keplerian_orbit_arc** for one sun-synchronous orbit starting at the *epoch* of the *start_event* which is a **space_mission_event_with_epoch**
- a **thermal_environment** which specifies the maximum solar flux, albedo and earthshine
- a **main_body_placement** which specifies an earthpointing attitude for the spacecraft

EXAMPLE 2 – The diagram below shows how **space_mission_event**s are used to define **space_mission_phase**s of **space_mission_aspect**s.

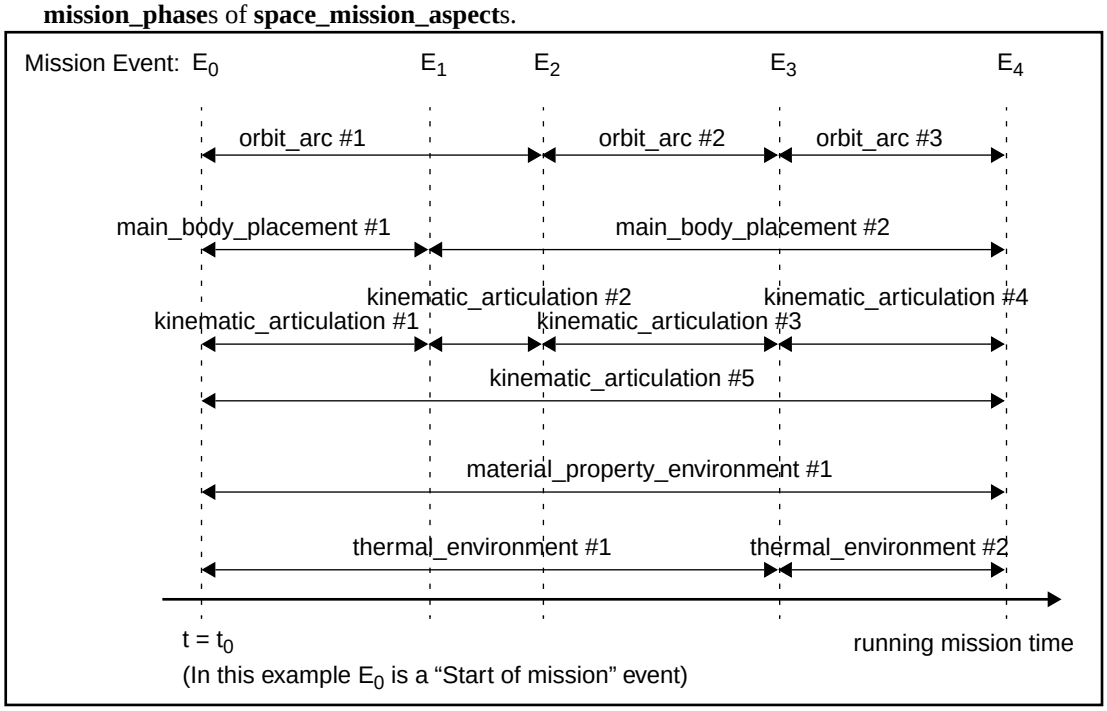


Figure 4–1 - Illustration of the use of space_mission_events for the specification of the different space_mission_phases for space_mission_aspects of a space_mission_sequence

4.2.8.3 space_mission_phase

A **space_mission_phase** is a type of **ato_phase** which specificies a certain number of **space_mission_aspect**s for the duration of an interval in a mission between a start and an end **space_mission_event**.

Attributes:

- *start_event:* reference to a **space_mission_event** that marks the start of the *space_aspects*
- *end_event:* reference to a **space_mission_event** that marks the end of the *space_aspects*
- *space_aspects*: a set of one or more references to **space_mission_aspect**s.

Assertions:

- The *of_case* shall be a **space_mission_case**
- *start_event* and *end_event* shall not point to the same **space_mission_event** instance
- The *of_case* of the *aspect* shall refer to the same **space_mission_case** as *of_case*
- If the *start_event* is a **space_mission_event_with_running_time** then the *end_event* shall also be a **space_mission_event_with_running_time**
- If the *start_event* is a **space_mission_event_with_true_anomaly** then the *end_event* shall also be a **space_mission_event_with_true_anomaly**
- Each of the *space_aspects* shall be a different type of **space_mission_aspect**

Notes:

NOTE 10 – The attributes and assertions are in addition to those inherited from **ato_phase**.

4.2.8.4 space_mission_aspect

A **space_mission_aspect** is a specification of a characteristic aspect of a mission. It is a generic object which captures the data common to all **space_mission_aspect**s and provides a generic referencing mechanism. A **space_mission_aspect** can be one of the following:

- an **orbit_arc**
- a **main_body_placement**
- a **kinematic_articulation**
- a **material_property_environment**
- a **thermal_environment**

Attributes:

- *of_case:* reference to a **space_mission_case**
- *id:* identifier string
- *name*: label string
- *description:* textual description

Assertions:

- The *id* shall be unique within the *of_case*

4.2.8.5 space_mission_event

A **space_mission_event** is a named event in a space mission. Such events are used to provide sequencing and synchronisation points in the mission timeline for the various **space_mission_aspect**s defined in the **space_mission_phase**s of a **space_mission_case**.

Attributes:

- *of_case:* reference to a **space_mission_case**
- *id:* identifier string
- *name*: label string
- *description:* text describing the event

Assertions:

- The *id* shall be unique within the *of_case*

4.2.8.6 space_mission_event_with_epoch

A **space_mission_event_with_epoch** is a type of **space_mission_event** for which the epoch is specified.

Attributes:

- *epoch:* reference to a calendar date and universal coordinated time

Notes:

NOTE 11 – The attributes and assertions are in addition to those of **space_mission_event**.

NOTE 12 – The *epoch* definition implies that the epoch is specified with year, month, day, hour, minute and second.

NOTE 13 – Calendar date and universal coordinated time are defined in the date_time_schema of [STEP-41]

4.2.8.7 space_mission_event_with_running_time

A **space_mission_event_with_running_time** is a type of **space_mission_event** for which a running time relative to some reference moment of time in the mission is specified.

Attributes:

- *running_time:* **time_measure** specifying the running mission time

Notes:

NOTE 14 – The attributes and assertions are in addition to those inherited from **space_mission_event**.

NOTE 15 – The *running_time* for a "Start of mission" event does not necessarily have to be set to zero. Any value – even a negative value – is allowed.

4.2.8.8 space_mission_event_with_true_anomaly

A **space_mission_event_with_true_anomaly** is a type of **space_mission_event** for which the true anomaly is specified relative to 0° true anomaly position, which defined as part of an **orbit_arc**.

Attributes:

- *true_anomaly:* **plane_angle_measure** specifying the true anomaly

Notes:

NOTE 16 – The attributes and assertions are in addition to those inherited from **space_mission_event**.

4.2.8.9 space_coordinate_system

A **space_coordinate_system** is a reference coordinate system used in the definition of orbit trajectories and spacecraft orientation. The following coordinate systems for space missions are defined (adapted from [Larson&Wertz], [ESABASE-RM], [THERMICA] and [ESARAD-RM]).

WORKING NOTE 3 – The basic systems 1) and 2) are considered in this version of the AP. The additional systems 3), 4), 5) and 6) are identified for a future version.

1) star_centric_inertial_ecliptic
   Origin at the centre of the star
   X axis – in the ecliptica and towards the specified standard direction
   Z axis – towards the ecliptic north pole, i.e. normal to the ecliptica
   Y axis – to complete the right-handed orthogonal coordinate system, i.e. $\vec{Y} = \vec{Z} \times \vec{X}$

   NOTE 17 – The standard direction is specified as the Gamma-50, the Gamma-2000 or the True-of-Date (TOD) vernal equinox.

2) planet_centric_star_fixed_equatorial
   Origin at the centre of the planet
   X axis – coincides with the "reference line" (see NOTE below)
   Z axis – towards the planet's north celestial pole (i.e. parallel to the south-to-north-pole vector)
   Y axis – to complete the right-handed orthogonal coordinate system. i.e. $\vec{Y} = \vec{Z} \times \vec{X}$

   NOTE 18 – The reference line is the orthogonal projection of the planet-centre-to-star-centre vector onto the planet's equatorial plane.

   NOTE 19 – The direction of the star is specified through the epoch or the declination. The right ascension of the star is 0° by definition.

3) planet_centric_inertial_ecliptic
   Origin at the centre of the planet
   X axis – parallel to the ecliptica and towards the specified standard direction
   Z axis – towards the ecliptic north pole
   Y axis – to complete the right-handed orthogonal coordinate system. i.e. $\vec{Y} = \vec{Z} \times \vec{X}$

NOTE 20 – The direction of the star is specified through the epoch or the right ascension.

4) planet_centric_inertial_equatorial
Origin at the centre of the planet
X axis – in the equatorial plane towards the specified standard direction
Z axis – towards the planet's north celestial pole (i.e. parallel to the south-to-north-pole vector)
Y axis – to complete the right-handed orthogonal coordinate system. i.e. $\vec{Y} = \vec{Z} \times \vec{X}$

NOTE 21 – The direction of the star is specified through the epoch or the right ascension and the declination.

5) planet_centric_star_fixed_ecliptic
Origin at the centre of the planet
X axis – parallel to the orthogonal projection of the centre-of-planet-to-true-of-date-star-position vector onto the ecliptica
Z axis – towards the ecliptic north pole
Y axis – to complete the right-handed orthogonal coordinate system. i.e. $\vec{Y} = \vec{Z} \times \vec{X}$

NOTE 22 – The direction of the star is specified through the epoch or the declination. The right ascension of the star is 0° by definition.

6) planet_centric_local
Origin at the centre of the planet
X axis – in the planet's equatorial plane and through the 0° longitude
Z axis – towards the planet's north celestial pole (i.e. parallel to the south-to-north-pole vector)
Y axis – to complete the right-handed orthogonal coordinate system. i.e. $\vec{Y} = \vec{Z} \times \vec{X}$

NOTE 23 – The direction of the star is specified through the epoch or through the right ascension and the declination.

Attributes:

- *reference_system*: one item from the enumeration:
  {"star_centric_inertial_ecliptic",
  "planet_centric_star_fixed_equatorial",
  "planet_centric_inertial_ecliptic",
  "planet_centric_inertial_equatorial"
  "planet_centric_star_fixed_ecliptic",
  "planet_centric_local" }
- *standard_direction:* OPTIONAL one item from the enumeration:
  {"gamma_50_vernal_equinox",
  "gamma_2000_vernal_equinox",
  "TOD_vernal_equinox"}

Assertions:

- *standard_direction* shall be specified when *reference_system* is one of
  {"star_centric_inertial_ecliptic",
  "planet_centric_inertial_ecliptic",
  "planet_centric_inertial_equatorial"}.

Notes:

NOTE 24 – The most often used reference system, for spacecraft orbiting the Earth, is "planet_centric_star_fixed_equatorial".

4.2.8.10 orbit_arc

An **orbit_arc** is a **space_mission_aspect** that captures the generic part of the data necessary for the specification of an orbit arc. Only its derived specialisations **discretised_orbit_arc** and **keplerian_orbit_arc** will be instantiated in the exchange dataset, not the **orbit_arc** itself.

Attributes:

- *orbit_class:* one item from {"general", "geostationary", "sun_synchronous", "molniya"}
- *governing_celestial_body:* reference to the **celestial_body** that governs the orbit
- *evaluation_events:* list of zero or more references to **space_mission_event**s, which specify the points in the orbit arc where evaluation shall take place

- *star_spherical_right_ascension:* OPTIONAL plane angle measure
- *star_spherical_declination:* OPTIONAL plane angle measure, which specifies the direction of the star
- *orbit_period:* OPTIONAL time measure with unit, specifying the period of the orbit

Assertions:

- The *evaluation_events* shall either all be **space_mission_event_with_epoch**s or all be **space_mission_event_with_running_time**s or all be **space_mission_event_with_true_anomaly**s
- If the *evaluation_events* are **space_mission_event_with_running_time**s the running time is relative to the *start_event* of *of_case*.
- If the *evaluation_events* are **space_mission_event_with_true_anomaly**s the true anomaly is relative to the 0° true anomaly
- The *running_time* or *epoch* or *true_anomaly* attribute values of the *evaluation_events* shall form a monotonic increasing list – i.e. the attribute value for an event shall be equal to or larger than the attribute value of its preceding event
- The *star_spherical_right_ascension* shall be specified if the *coordinate_system* of the **space_mission_case** is "planet_centric_star_fixed_ecliptic" or "planet_centric_star_fixed_equatorial", otherwise it shall not be specified.
- –360° ≤ *star_spherical_right_ascension* ≤ 360°
- –90° ≤ *star_spherical_declination* ≤ 90°

Notes:

NOTE 25 – The attributes and assertions are in addition to those inherited from **space_mission_aspect**

NOTE 26 – The **space_coordinate_system** for the **orbit_arc** is specified in the *coordinate_system* of *of_case*.

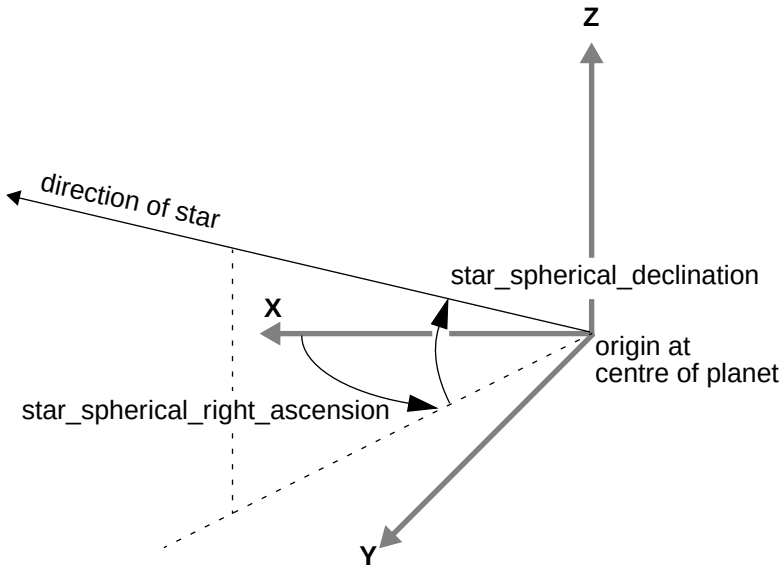


Figure 4–2 - Illustration of the definitions of the star_spherical_right_ascension and star_spherical_declination angles

Examples:

EXAMPLE 3 – "SunSync-Cold" is an example of such a *name* of an **orbit_arc**.

EXAMPLE 4 – An example for *description* is:
"Sun-sync, 22:30LST, h=max=800km".

EXAMPLE 5 – Another example for *description* is:
"STS parking orbit, h=350km, I=28.5°, beta=52° (maximum)"

4.2.8.11 discretised_orbit_arc

A **discretised_orbit_arc** is a specialisation of an **orbit_arc**, in which a general orbit arc is defined as an ephemeris – i.e. as a list of positions with velocity vectors. This enables support of complex orbits – e.g. perturbed orbits. For informational purposes a set of Kepler orbit parameters may be referenced to indicate an associated unperturbed orbit arc.

Attributes:

- *centre_of_governing_celestial_body*: reference to a **cartesian_point**
- *positions_and_velocities:* list of **orbit_position_and_velocity** objects
- *orbit_generator_name:* label string providing information on the orbit generator package and/or algorithm, which was used to generate the orbit evaluation positions
- *kepler_parameters:* OPTIONAL reference to a **kepler_parameter_set**

Assertions:

- There shall be one *positions_and_velocities* entry for each *evaluation_event* entry (see the definition in subclause 4.2.8.10 “orbit_arc”)

Notes:

NOTE 27 – The attributes and assertions are in addition to those inherited from **orbit_arc**.

NOTE 28 – The *centre_of_governing_celestial_body* and the *positions_and_velocities* are expressed in the *coordinate_system* of the **space_mission_case** to which the **orbit_arc** belongs.

NOTE 29 – The 0° true anomaly is defined by the vector from the *centre_of_governing_celestial_body* to the first orbit position in the *positions_and_velocities* list.

4.2.8.12 keplerian_orbit_arc

A **keplerian_orbit_arc** is a type of **orbit_arc** which defines an undisturbed basic Keplerian orbital arc.

Attributes:

- *kepler_parameters:* reference to a **kepler_parameter_set**, which specifies a set of 6 classical Kepler orbit parameters

Assertions:

- None

Notes:

NOTE 30 – The attributes and assertions are in addition to those inherited from **orbit_arc**.

4.2.8.13 kepler_parameter_set

A **kepler_parameter_set** is an object collecting the 6 classical Kepler orbit parameters.

Attributes:

- *semi_major_axis:* a positive length measure
- *eccentricity:* a positive real number, which specifies the orbit trajectory shape
  - *eccentricity* = 0: circular orbit
  - 0 < *eccentricity* < 1: elliptical orbit
  - *eccentricity* = 1: parabolic orbit
  - *eccentricity* > 1: hyperbolic orbit
- *inclination:* plane angle measure between XY-plane of the *coordinate_system* and the orbit plane, –180° ≤ angle ≤ 180°
- *right_ascension_of_ascending_node:* plane angle measure, –360° ≤ angle ≤ 360°
- *argument_of_periapsis:* plane angle measure from ascending node to periapsis, 0° ≤ angle < 360°
- *true_anomaly_at_start:* plane angle measure from periapsis to the start position of the orbit arc, –360° ≤ angle ≤ 360°

Assertions:

- If *reference_system* of *coordinate_system* is “star_centric_inertial” or “planet_centric_inertial”, then the right ascension of the ascending node is measured relative to the *standard_direction*.

- If *reference_system* of *coordinate_system* is “planet_centric_local”, then the right ascension of the ascending node is measured from the reference line, which is the projection of the planet-to-star vector on the equatorial plane of the planet.
- If the first element of the *evaluation_events* list is a **space_mission_event_with_true_anomaly** then its *true_anomaly* shall be equal to the *true_anomaly_at_start*

Notes:

NOTE 31 – The 0° true anomaly is defined by the vector from the centre of the governing celestial body to the periapsis of the orbit

NOTE 32 – Figures 4–3, 4–4, 4–5 and 4–6 illustrate the definition of the Kepler orbit parameters.

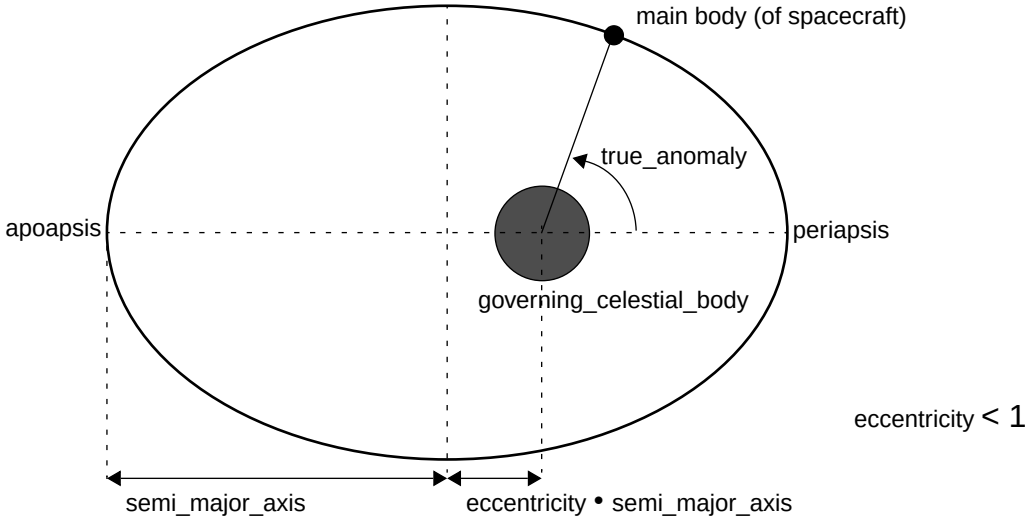


Figure 4–3 - Illustration of the parameters for a basic elliptical Keplerian orbit

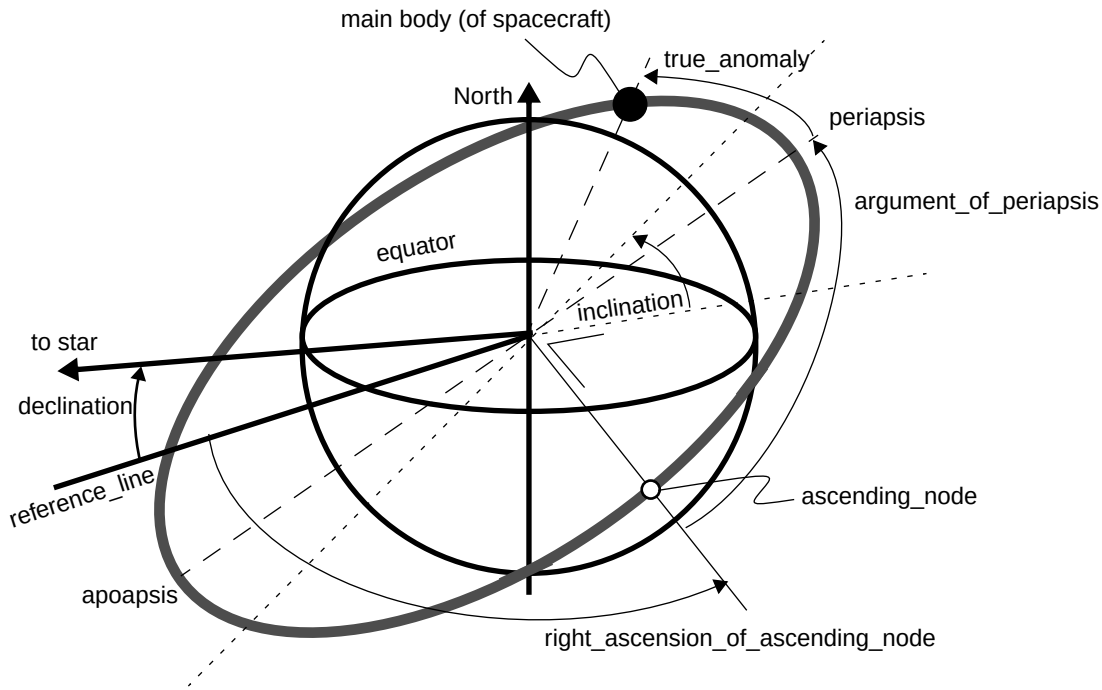


Figure 4–4 - Illustration of parameters for a Keplerian elliptical orbit around a planet, in the typical "planet_centric_star_fixed_equatorial" space coordinate system

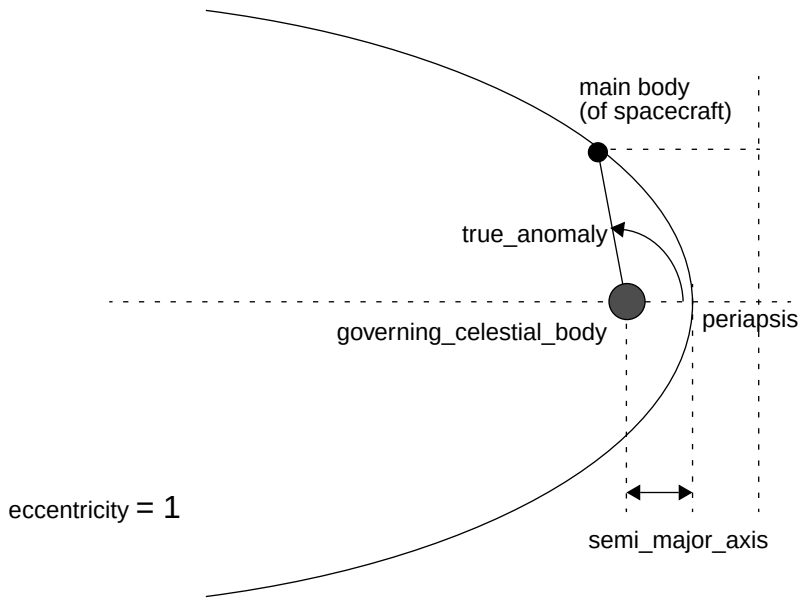


Figure 4–5 - Illustration of parameters for a Keplerian parabolic orbit

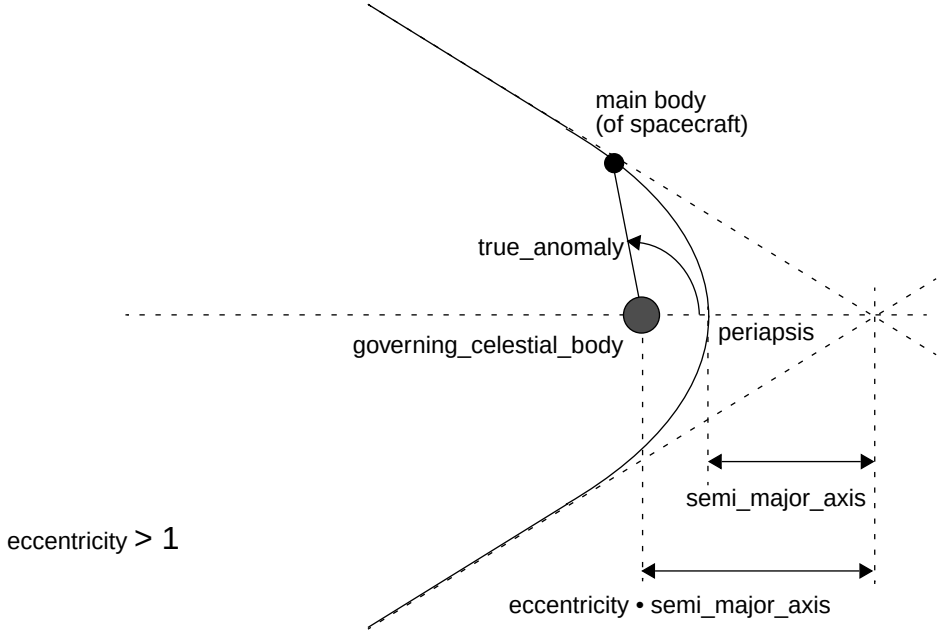


Figure 4–6 - Illustration of parameters for a Keplerian hyperbolic orbit

4.2.8.14 orbit_position_and_velocity

An **orbit_position_and_velocity** is the specification of an orbit position and a velocity vector. This is an element of the ephemeris of an orbit. The position and velocity are defined in the *coordinate_system* of the *of_case* of the **discretised_orbit_arc** which references the **orbit_position_and_velocity.**

Attributes:

- *position:* reference to a **cartesian_point**
- *velocity_magnitude:* velocity measure, which specifies the magnitude of velocity at *position*.
- *velocity_direction:* direction of the velocity vector, defined by three Cartesian direction component velocities at *position*.

4.2.8.15 celestial_body

A **celestial_body** is the specification of the name and characteristic gravitational attributes of a celestial body.

Attributes:

- *name:* label string – e.g. “Sun”, “Earth”, “Jupiter”, “Moon”
- *radius:* reference to a **length_measure** or to a **function_valued_property** that defines the radius of the celestial body. In the first case, the radius is a constant, in the second it shall be a function of the latitude and/or longitude of the celestial body.
- *gravitational_constant:* a measure defined as the product of the Newtonian constant of gravitation ($G = 6.67259 \cdot 10^{-11}\ m^3 kg^{-1} s^{-1}$) and the mass of the celestial body, its dimension is $L^3 \cdot T^{-2}$
- *sidereal_rotation_period:* OPTIONAL time measure

Assertions:

- *name* shall be unique within the exchange dataset
- The following list of celestial body *name*s shall be pre-defined: {“Sun”, “Mercury”, “Venus”, “Earth”, “Mars”, “Jupiter”, “Saturn”, “Neptune”, “Uranus”, “Pluto”, “Moon”}.

Notes:

NOTE 33 – If one of the pre-defined names is used, a software package receiving an exchange dataset may apply default data it contains on the specific celestial body.

Examples:

EXAMPLE 6 – For Earth the *gravitational_constant* is $3.986005 \cdot 10^{14}$ m^3/s^2

EXAMPLE 7 – For Sun the *gravitational_constant* is $1.32712438 \cdot 10^{20}$ m^3/s^2

EXAMPLE 8 – For Earth the *sidereal_rotation_period* is 23 hours 56 minutes 4 seconds = 86164 s

4.2.8.16 celestial_body_with_orbit

A **celestial_body_with_orbit** is a type of **celestial_body** to which the specification of the orbit around its governing celestial body is added. This enables the definition of a complex orbital system with a chain of celestial bodies and orbits.

Attributes:

- *orbit:* reference to an **orbit_arc**, which specifies the orbit of the celestial body around its governing celestial body; the reference plane for orbit inclination is the ecliptica.

Assertions:

- The object shall only be used when the *coordinate_system* of the **space_mission_case** to which the *orbit* belongs, is one of the "inertial" systems.

Notes:

NOTE 34 – The attributes and assertions are in addition to those inherited from **celestial_body**.

Examples:

EXAMPLE 9 – A planet in its orbit around the Sun.

EXAMPLE 10 – The Moon in its orbit around the Earth, and the Earth in its orbit around the Sun.

4.2.8.17 thermal_aspects_of_celestial_body

A **thermal_aspects_of_celestial_body** specifies the thermal aspects of a celestial body.

Attributes:

- *of_celestial_body:* reference to **celestial_body**
- *black_body_surface_temperature*: reference to a **thermodynamic_temperature_measure** or to a **function_valued_property** that defines the equivalent black body temperature of the surface of the *of_celestial_body.* In the first case, the temperature is a constant, in the second it shall be a function of the latitude and/or longitude of the *of_celestial_body*
- *notional_distance_to_celestial_body:* OPTIONAL **positive_length_measure** that specifies the distance to the celestial body to be used for calculation of the thermal flux coming from the *of_celestial_body.*
- *spectrum:* OPTIONAL reference to a **thermal_radiation_spectrum**
- *albedo_reflectors:* set of zero or more references to **albedo_reflector**s

Notes:

NOTE 35 – The thermal radiation flux from a celestial body can be calculated using the formula: $\Phi = \sigma T^4 \cdot \frac{r^2}{d^2}$, where: Φ is flux in e.g. [W/m^2], σ is the Stefan-Boltzmann constant, T is the black_body_surface_temperature, r is the radius of the celestial_body and d is the distance from the centre of the celestial body.

NOTE 36 – Use of the notional distance enables the specification of a solar (or other stellar) thermal environment without defining the *orbit* of a planet explicitly in a **celestial_body_with_orbit** instance.

NOTE 37 – The default is Planck's black body thermal radiation curve for the *black_body_surface_temperature* – e.g. the predefined "Generic infra-red" **thermal_radiation_spectrum** for Earth.

NOTE 38 – Planck's black body thermal radiation curve is $E(T) = \int_{\lambda_1}^{\lambda_2} \frac{2\pi hc^2}{\lambda} \cdot \frac{1}{e^{ch/(k\lambda T)} - 1} d\lambda$

where: E is emitted energy per unit area for the wavelength range λ_1 to λ_2, h is Planck's constant, c is the speed of light, λ is wavelength, k is Boltzmann's constant and T is the black body temperature.

Examples:

EXAMPLE 11 – For a spacecraft near Earth the mean Sun-Earth distance ($1.49675 \cdot 10^{11}$ m) is often used as a notional distance.

4.2.8.18 albedo_reflector

An **albedo_reflector** is the specification of thermal radiation through albedo reflection off a celestial body. The albedo reflection factor quantifies the fraction of incident thermal radiation from a star that is diffusely reflected by a planet.

Attributes:

- *albedo_originator:* reference to a **thermal_aspects_of_celestial_body**, which defines the celestial body from which the albedo radiation originates
- *albedo_reflection_factor:* **ratio_measure** or reference to a **function_valued_property** that defines the albedo reflection factor. In the first case, the albedo reflection factor is a constant, in the second, it shall be a function of the latitude and longitude of the referencing **celestial_body** (through the *of_celestial_body* attribute of the **thermal_aspects_of_celestial_body** that references the **albedo_reflector**).
- *spectrum:* OPTIONAL reference to a **thermal_radiation_spectrum**

Assertions:

- The value of the *albedo_reflection_factor* shall be within the range [0., 1.]
- The *albedo_originator* shall not reference the same **thermal_aspects_of_celestial_body** for which an **albedo_reflector** is being defined

Notes:

NOTE 39 – If *spectrum* is left undefined, it is considered to be the same as the emission **thermal_radiation_spectrum** of *albedo_originator*

4.2.8.19 main_body_placement

A **main_body_placement** is a type of **space_mission_aspect** which specifies the position and orientation of the main body of an engineering object, in space.

The position is specified as a translation relative to a point on the main body's trajectory, i.e. the orbit position at an actual evaluation event.

The orientation is specifed by one or two pointing directions for, respectively, one or two axes of the object, optionally followed by a sequence of axis rotations.

The main body of an engineering object is represented by a root **geometric_model** – i.e. a single **geometric_model** when no model/submodel tree is present and the root level **geometric_model** of a model/submodel tree when such a tree is present. A geometric model/submodel tree and submodel occurrences are defined through **geometric_submodel_usage**s (see subclause 4.2.9.3 on page 4-24).

The local coordinate system of a main body is the coordinate system of the associated **geometric_model**.

In further specialisations of **main_body_placement** a constant angular velocity or spinning about an axis can be specified, using one of the following application objects:

- a **main_body_placement_with_constant_rotation**;
- a **main_body_placement_with_spinning**.

Attributes:

- *translation:* OPTIONAL Cartesian translation, specifying a translation of the origin of the coordinate system of the main body. The translation is specified with respect to a reference point in space, which would normally be an evaluation point on an orbit arc.

- *primary_orientation:* OPTIONAL reference to a **pointing_in_space**, specifying a primary pointing of the main body for the *pointer* defined in the **pointing_in_space**
- *secondary_orientation:* OPTIONAL reference to a **pointing_in_space**, specifying a secondary pointing of the main body for the *pointer* defined in the **pointing_in_space**, while fulfilling the constraint of the *primary_orientation*
- *rotations:* list of zero or more references to **attitude_rotation**s, specifying a sequence of rotations to be applied after the primary and secondary orientations

Assertions:

- When both *primary_orientation* and *secondary_orientation* are left undefined then the initial orientation is assumed to be according to the **space_coordinate_system** of the **space_mission_case**, i.e. the main body X-axis along **space_coordinate_system** X-axis, etc.
- The *secondary_orientation* shall not be defined when the *primary_orientation* is undefined
- The *pointer* of the *secondary_orientation* shall be different from the one specified for the *primary_orientation*.
- The *pointing_direction* of the *secondary_orientation* shall not be parallel or anti-parallel to the *pointing_direction* of the *primary_orientation*.

Notes:

NOTE 40 – The attributes and assertions are in addition to those inherited from **space_mission_aspect**.

NOTE 41 – The orientation of a spacecraft is often called “attitude”.

NOTE 42 – While the primary orientation is an actual orientation that can always be executed perfectly, the secondary orientation defines, in principle, an approximate orientation. The actual secondary orientation is defined as the direction for which the deviation between the specified *desired_pointing* and the *pointer* of the *secondary_direction* is minimised under the constraint of the correct execution of the *primary_orientation*.

4.2.8.20 main_body_placement_with_constant_rotation

A **main_body_placement_with_constant_rotation** is a **main_body_placement** to which the specification of a rotation with constant angular velocity about a given axis is added.

Attributes:

- *rotation_axis*: reference to an **axis_placement**, which specifies the rotation axis. The rotation axis passes through the *location* of the **axis_placement** and its direction is defined by the *z_axis_direction* of the **axis_placement**.
- *angular_velocity:* specifies a constant angular velocity measure with dimension [plane angle / unit of time]

Assertions:

- None

Notes:

NOTE 43 – The attributes and assertions are in addition to those inherited from **main_body_placement**.

NOTE 44 – A positive value for *angular_velocity* indicates a positive sense of rotation in a right-handed orthogonal coordinate system.

4.2.8.21 main_body_placement_with_spinning

A **main_body_placement_with_spinning** is a **main_body_placement** to which the specification of spinning about a given axis of a main body is added. When applicable, also the number of spin positions that should be used for averaging during thermal-radiative flux computations are specified.

Attributes:

- *spin_axis*: reference to an **axis_placement**, which specifies the rotation axis. The rotation axis passes through the *location* of the **axis_placement** and its direction is defined by the *z_axis_direction* of the **axis_placement**.

- *number_of_spin_positions:* OPTIONAL INTEGER that specifies the number of positions per 360° spin revolution to be evaluated for spin averaging in thermal radiation flux computations, for each orbit evaluation position.

Assertions:

- The *number_of_spin_positions* shall be a positive integer

Notes:

NOTE 45 – The attributes and assertions are in addition to those inherited from **main_body_placement**.

NOTE 46 – Spinning is a relatively fast rotation, in the following sense: the duration of one spin revolution is short with respect to the typical thermal response times for equipment of the involved object, in particular of exposed parts on its exterior.

4.2.8.22 pointing_in_space

A **pointing_in_space** specifies a direction in space, which can be used for orientating objects in space. A *pointer* direction for an object and a *desired_pointing* direction in space are specified. The object is perfectly pointed when the *pointer* is pointing in the *desired_pointing* direction.

Attributes:

- *pointer*: a 3D Cartesian direction, which specifies a direction vector in the local coordinate system of the object to be pointed. The pointer starts in the origin of the local coordinate system.
- *desired_pointing:* one item from the enumeration:
  {governing_star, equatorial_projection_of_governing_star, vernal_equinox, ecliptic_north_pole, planet_centre, planet_zenith, planet_north_pole, angular_momentum, spacecraft_velocity_vector, reference to a **general_star_pointing**, reference to a **general_space_pointing**}

Notes:

NOTE 47 – In search for a complete, but still not overly complex, coverage of the different possibilities to specify a pointing for attitude definition in space, the pointing directions defined below were selected. This extensive method of orientation definition is preferred over a more concise mathematical method – using e.g. bare transformation matrices and “hard-defined” initial pointings – because such an extensive method captures a lot of mission information, that would otherwise be lost. Such information is very useful for human interpretation purposes. Inputs from [Larson&Wertz], [ESABASE-RM], [THERMICA] and [ESARAD-RM] were used to compile this definition.

The following eleven pointing directions are provided to be used for orientation in space:

a) governing_star
   in the true direction to the governing star at the current epoch
b) equatorial_projection_of_governing_star
   parallel to the normal projection of the planet-centre-to-governing-star vector on the equatorial plane of the planet (this projection line is commonly called the “reference line”)
c) vernal_equinox
   towards the vernal equinox direction
d) ecliptic_north_pole
   towards the north pole of the ecliptic plane of the star-planet system
e) general_star
   in a direction specified by a right ascension and a declination relative to the vernal equinox
f) planet_centre
   towards the centre of the planet
g) planet_zenith
   away from the centre of the planet, i.e. the opposite direction of planet_centre
h) planet_north_pole
   parallel to the south-to-north-pole vector of the planet
i) angular_momentum
   parallel to the positive normal on the orbit plane

j) spacecraft_velocity_vector
parallel to the spacecraft's velocity vector, i.e. tangent to its orbit trajectory

k) general_direction
parallel to the direction defined by means of a direction vector in a given referenced **space_coordinate_system**

NOTE 48 – The **general_star_pointing** and **general_space_pointing** need additional parameters, see their respective definitions.

4.2.8.23 general_star_pointing

A **general_star_pointing** specifies a pointing direction to an identified star.

Attributes:

- *of_case*: reference to a **space_mission_case**
- *star_name:* label string, specifying the name of the star for information only
- *star_spherical_right_ascension:* **plane_angle_measure** from the X axis in the XY plane to the plane defined by the Z axis and the vector to the centre of the star
- *star_spherical_declination:* **plane_angle_measure** from the XY plane in the plane defined by the Z axis and the vector to the centre of the star.

Assertions:

- The *star_name* shall be unique within the *of_case*
- -360° ≤ *right_ascension* ≤ 360°
- -90° ≤ *declination* ≤ 90°

Notes:

NOTE 49 – The pointing is defined in the **space_coordinate_system** referenced by the *of_case*

NOTE 50 – See also the illustration in Figure 4–2 on page 4-12.

4.2.8.24 general_space_pointing

A **general_space_pointing** specifies a general pointing direction in the space coordinate system of a **space_mission_case**.

Attributes:

- *of_case*: reference to a **space_mission_case**
- *name*: label string
- *direction:* a 3D Cartesian direction vector, which specifies the pointing direction in the *coordinate_system* of the *of_case*

Assertions:

- The *name* shall be unique within the *of_case*

Notes:

NOTE 51 – The pointing is defined in the **space_coordinate_system** referenced by the *of_case*

4.2.8.25 attitude_rotation

An **attitude_rotation** is a specification of a rotation about a given axis of a product.

Attributes:

- *axis:* the specification of the axis of rotation, which is defined by a 3D Cartesian direction vector in the local coordinate system of an engineering object. The axis passes through the origin of the local coordinate system.
- *angle:* plane angle measure or reference to a **function_valued_property** that specifies the angle of rotation. In the first case, it is a constant, in the second it shall be a function of one parameter from {orbit_computation_time, epoch or true_anomaly}. A positive value indicates a positive rotation sense in a right-handed orthogonal coordinate system.

Assertions:

- -360° ≤ *angle* ≤ 360°

Notes:

NOTE 52 – Some software tools in the application domain only have mathematical transformation matrices available for specification of attitude rotations. In this case no information on a sequence of rotations about principal axes is known. A STEP-TAS export module for such a tool shall use the following pseudo-algorithm to derive a sequence of rotations about principal axes:

- Try to derive a rotation about the X-axis with: -180° < rotation-angle ≤ 180°
- Try to derive a rotation about the Y-axis with: -180° < rotation-angle ≤ 180°
- Try to derive a rotation about the Z-axis with: -180° < rotation-angle ≤ 180°

Any 3-dimensional rotation matrix can be decomposed into a sequence of two rotations about two principal axes.

4.2.8.26 kinematic_articulation

A **kinematic_articulation** is a **space_mission_aspect**, which specifies the actual articulation of a **kinematic_pair** (and therefore a **kinematic_joint**) during an interval in a space mission. The start and end of the interval are specified in the **space_mission_phase** or **space_mission_phase**s which reference the **kinematic_articulation**.

Attributes:

- *start_pair_value*: reference to a **kinematic_pair_value**, which specifies the initial value
- *end_pair_value*: reference to a **kinematic_pair_value**, which specifies the final value

Assertions:

- *start_pair_value* and *end_pair_value* shall reference **kinematic_pair_value**s for the same **kinematic_pair** (referenced by the *applies_to_pair* attribute of **kinematic_pair**)

Notes:

NOTE 53 – The attributes and assertions are in addition to those inherited from **space_mission_aspect**.

4.2.8.27 thermal_environment

A **thermal_environment** is a type of **space_mission_aspect** that captures the parameters that define an external thermal environment. It can be used for space environments as well as for test chamber environments.

In typical LEO or GEO missions there are two **thermal_aspects_of_celestial_body** objects, “Sun” and “Earth”. It is entirely possible however to allow for more celestial bodies with their thermal characteristics to be relevant to a space mission.

Attributes:

- *environmental_sink_name:* label string
- *environmental_sink_node_id:* OPTIONAL integer thermal node id
- *environmental_sink_temperature:* far field temperature measure
- *celestial_body_environment:* list of zero or more **thermal_aspects_of_celestial_body** objects
- *star_flux_constant:* OPTIONAL thermal radiation flux measure, which specifies a constant thermal flux coming from a star, with the dimension of unit power per unit area

Assertions:

- The list entries in *celestial_body_environment* must reference different **celestial_body**s
- If *star_flux_constant* is specified and the *celestial_body_environment* list contains a **thermal_aspects_of_celestial_body** for a star, than *star_flux_constant* overrides the flux that can be derived from *black_body_surface_temperature* and *notional_distance_from_celestial_body* of the referenced **thermal_aspects_of_celestial_body** instance(s)

Examples:

EXAMPLE 12 – Typical *environmental_sink_name*s are “Space”, “Deep Space” or “Shroud”

EXAMPLE 13 – A typical *environmental_sink_node_id* is 9999, which is often used for “Deep Space”

EXAMPLE 14 – A typical value for *environmental_sink_temperature* equals 4K for deep space

EXAMPLE 15 – For *star_flux_constant* the year-averaged Solar Constant near Earth (1371 W/m^2) is often used.

4.2.8.28 thermal_radiation_spectrum

A **thermal_radiation_spectrum** is a named spectrum for thermal electromagnetic radiation defining the radiation intensity as a function of wavelength.

Attributes:

- *name:* label string
- *description:* text, describing details on e.g. validity and/or application of the spectrum
- *values:* reference to a **function_valued_property** that defines a *normalised_intensity* property class as a function of *wave_length*

Examples:

EXAMPLE 16 – Spectra could be specified for: “Solar”, “ESTEC-LSS”, “Blackbody_100K”, “Earth-Albedo”

4.2.8.29 operational_mode

WORKING NOTE 4 – This item is identified for a future version and not in scope for the AIM.

4.2.8.30 degradation_dose_profile

WORKING NOTE 5 – This item is identified for a future version and not in scope for the AIM.

4.2.9 geometric_model UoF

4.2.9.1 geometric_context

A **geometric_context** specifies all global assignments for use in the **geometric_mode**ls contained in an exchange dataset.

Attributes:

- *length_unit:* reference to a **length_unit**, which specifies the actual length unit for all **length_measure**s in the geometric models

  EXAMPLE 17 – Typically metre or millimetre would be used.
- *plane_angle_unit:* reference to a **plane_angle_unit**, which specifies the actual plane angle unit for all **plane_angle_measure**s in the geometric models.

  EXAMPLE 18 – Typically degree would be used in space applications.
- *point_in_plane_uncertainty*: reference to a **length_measure**, denoting the maximum distance a fourth or higher definitional point of a planar geometric shape may deviate from the plane defined by the first three definitional points.
- *point_coincidence_uncertainty*: reference to a **length_measure**, denoting the maximum distance two points may be apart while still being considered to define the same point in a coordinate space.

Assertions:

- There shall be one and only one **geometric_context** per exchange dataset

4.2.9.2 geometric_model

A **geometric_model** is a model representing the shapes of a product and the shapes of its (intended) operating environment, as far as these are relevant for analysis purposes.

Attributes:

- *id:* identifier
- *name:* label
- *description:* text
- *surfaces*: one or more references to **primitive_bounded_surface**s, **compound_surface_shell**s, **boolean_construction_surface**s or **boolean_difference_surface**s
- *submodel_usages*: zero or more references to **geometric_submodel_usage**s

Assertions:

- None

Notes:

NOTE 54 – In this protocol the supported shapes are restricted to bounded surfaces.

4.2.9.3 geometric_submodel_usage

A **geometric_submodel_usage** is a specification of the usage of one **geometric_model** as a submodel of another **geometric_model**, which is then called its 'supermodel'. It has an axis placement attribute which makes it possible to define a new coordinate system for the submodel with respect to the coordinate system of its supermodel.

The **geometric_model** instances in a model/submodel tree are considered to be rigid bodies. Only at the supermodel/submodel interfaces relative (kinematic) motion is possible.

Attributes:

- *id:* identifier
- *role:* label specifying the usage of the *submodel* in the context of its supermodel
- *submodel*: reference to a geometric_model
- *placement:* reference to an **axis_placement**, which specifies the position and orientation of the coordinate system for the *submodel* with respect to the coordinate system of its supermodel

Assertions:

- The model/submodel tree established by **submodel_usage** shall form an acyclic graph

Notes:

NOTE 55 – In the **space_mission_phase** object it is possible to define dynamic position and orientation for the root geometric model through a **main_body_placement** referenced in the *space_aspects* attribute. In addition, dynamic position and orientation for submodels can be defined through a **kinematic_articulation** also referenced in the *space_aspects* attribute. In contrast the position and orientation defined above in *placement* is used for static construction of the model.

4.2.9.4 geometric_construction_point

A **geometric_construction_point** is a definition of a named point in 3-D space in Cartesian, orthonormal, right-handed coordinate system with x, y and z coordinates. A **geometric_construction_point** can be referenced at any place in a shape definition where a point is demanded. The unit of length is specified in the **geometric_context** (see subclause 4.2.9.1).

Attributes:

- *id:* identifier string
- *coordinates:* list of three **length_measure**s, for x, y and z coordinates respectively

Assertions:

- The *id* shall be unique within the **geometric_model** in which the point is used

4.2.9.5 axis_placement

An **axis_placement** defines the location and orientation of a Cartesian orthonormal coordinate system in three-dimensional space with respect to the coordinate system in which it is specified.

Attributes:

- *location:* reference to a **geometric_construction_point**
- *z_axis_direction:* exact direction definition of the z-axis
- *x_axis_direction:* (approximate) direction definition of the x-axis

The exact direction of the x-axis is defined as orthogonal to the *z_axis_direction*, in the plane formed by the *x_axis_direction* and *z_axis_direction* and as close as possible to the given *x_axis_direction*

The direction of the y-axis is derived from the vectorproduct of the z-axis and x-axis directions, to form a right-handed orthonormal coordinate system.

4.2.9.6 Surface definitions

In this protocol a **geometric_model** is constructed from surfaces. In order to unambiguously specify the various surface objects, first the general definitions below are given.

- A bounded surface is a surface of finite area with identifiable boundaries.
- An unbounded surface is a surface that extends infinitely in one or more directions.
- The following surface categories are identified:
  - Primitive bounded surfaces,
    which are the elementary surfaces such as triangles, rectangles, discs, cones, cylinders, etc.
  - Compound bounded surfaces,
    which are surfaces that consist of a named group of connected primitive bounded surfaces.
  - Auxiliary boolean construction surfaces,
    which are unbounded surfaces that may be used in the specification of boolean results surfaces.
  - Boolean result surfaces,
    which are surfaces that are created by applying a boolean operation on two existing surfaces.
- The positive side of a surface is the side that is identified by the positive geometric normal, which is given as part of the definition of all primitive bounded surfaces and auxiliary boolean construction surfaces.
- The sense of a surface is the side that is used in boolean construction operations. When the sense is positive, the solid at the positive side of the surface is used as a boolean operand and when the sense is negative the opposite side of the surface is used as a boolean operand.

4.2.9.7 Primitive bounded surfaces

This subclause defines the primitive bounded surfaces:

- **primitive_bounded_surface**,
- **triangle**,
- **rectangle**,
- **quadrilateral**,
- **disc**,
- **cylinder**,
- **cone**,
- **sphere**,
- **paraboloid**.

4.2.9.7.1 primitive_bounded_surface

A **primitive_bounded_surface** is a surface of finite area with identifiable boundaries. It is a generic object which specifies the attributes common to all primitive bounded surfaces.

Attributes:

- *id:* identifier string
- *placement:* OPTIONAL reference to an **axis_placement,** that specifies a local coordinate system in which the positioning attributes of the subtypes of **primitive_bounded_surface** are expressed.
- *colour*: OPTIONAL reference to a **colour_rgb**, which specifies a colour that may be used for presentation purposes

4.2.9.7.2 triangle

A **triangle** is a planar **primitive_bounded_surface** defined by three points.

NOTE 56 – The attributes and assertions are in addition to those inherited from **primitive_bounded_surface**.

Attributes:

- *p1:* reference to a **geometric_construction_point**
- *p2:* reference to a **geometric_construction_point**
- *p3:* reference to a **geometric_construction_point**

The positive normal on the surface is defined by $\overrightarrow{p1p2} \times \overrightarrow{p1p3}$

The first parametric direction (direction-1) is defined along $\overrightarrow{p1p2}$ vector,

the second parametric direction (direction-2) is defined along the the $\overrightarrow{p2p3}$ vector.

A sketch of the definition is given in the figure below.

Assertions:

- None

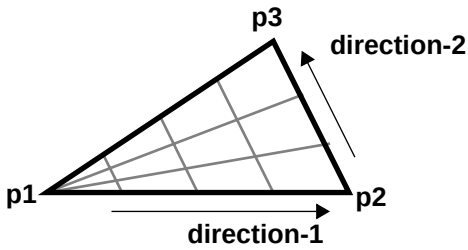


Figure 4–7 - Sketch of triangle

NOTE 57 – The dashed lines in the sketch illustrate the way in which the surface is (uniformly) meshed into multiple faces by setting a number of mesh divisions for each of the parametric directions. See subclause 4.2.11.4 on page 4-45 for its application. In Figure 4–7 the number of mesh divisions in the first parametric direction is set to 4, while the number of mesh divisions in the second parametric direction is set to 3.

4.2.9.7.3 rectangle

A **rectangle** is a planar rectangular **primitive_bounded_surface** defined by three points.

NOTE 58 – The attributes and assertions are in addition to those inherited from **primitive_bounded_surface**.

Attributes:

- *p1:* reference to a **geometric_construction_point**
- *p2:* reference to a **geometric_construction_point**
- *p3:* reference to a **geometric_construction_point**

The positive normal on the surface is defined by $\overrightarrow{p1p2} \times \overrightarrow{p1p3}$

The first parametric direction (direction-1) is defined along the $\overrightarrow{p1p2}$ vector,

the second parametric direction (direction-2) is defined along the $\overrightarrow{p1p3}$ vector.

A sketch of the definition is given in the figure below.

Assertions:

- The angle between $\overrightarrow{p1p2}$ and $\overrightarrow{p1p3}$ shall be 90°

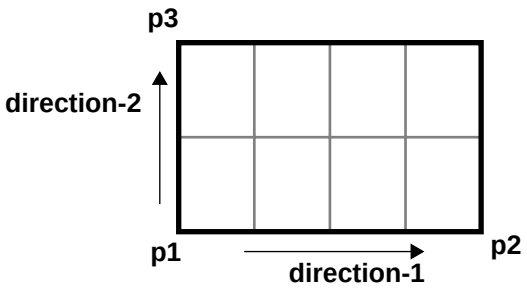


Figure 4–8 - Sketch of rectangle

4.2.9.7.4 quadrilateral

A **quadrilateral** is a planar **primitive_bounded_surface** defined by four points, with linear edges.

NOTE 59 – The attributes and assertions are in addition to those inherited from **primitive_bounded_surface**.

Attributes:

- *p1:* reference to a **geometric_construction_point**
- *p2:* reference to a **geometric_construction_point**
- *p3:* reference to a **geometric_construction_point**
- *p4:* reference to a **geometric_construction_point**

The positive normal on the surface is defined by $\overrightarrow{\mathrm{p1p2}} \times \overrightarrow{\mathrm{p1p3}}$

The first parametric direction (direction-1) is defined along $\overrightarrow{\mathrm{p1p2}}$ vector,

the second parametric direction (direction-2) is defined along the the $\overrightarrow{\mathrm{p1p3}}$ vector.

A sketch of the definition is given in the figure below.

Assertions:

- Each internal angle of the quadrilateral shall be smaller than 180°.

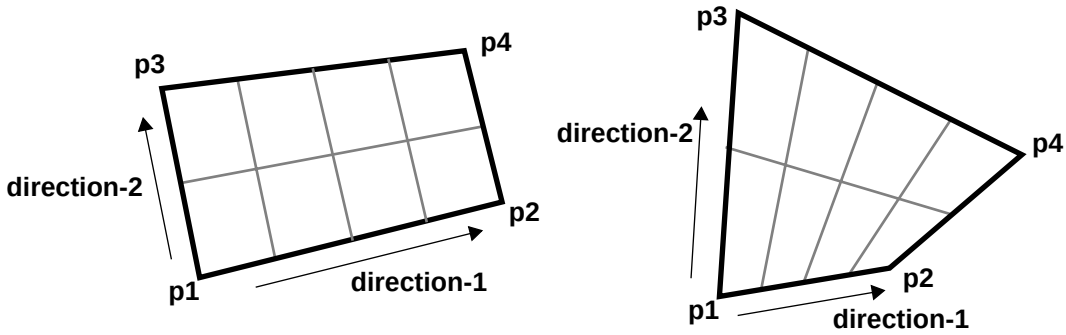


Figure 4–9 - Sketch of two examples of quadrilaterals

4.2.9.7.5 disc

A **disc** is a **primitive_bounded_surface**, which defines a disc, ring, sector of disc or sector of ring.

NOTE 60 – The attributes and assertions are in addition to those inherited from **primitive_bounded_surface**.

Attributes:

- *p1:* reference to a **geometric_construction_point**
  centrepoint of the disc and first point of the axis of revolution
- *p2:* reference to a **geometric_construction_point**
  second point of the axis of revolution
- *external_diameter:* length_measure
- *internal_diameter:* length_measure
- *p3:* reference to a **geometric_construction_point** to define the p1p2p3 reference plane for the *start_angle* and *end_angle*
- *start_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the start of a sector
- *end_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the end of a sector

The positive normal on the surface is defined by $\overrightarrow{p1p2}$

The first parametric direction (direction-1) is defined along the radius of the disc, the second parametric direction (direction-2) is defined along the positive direction of revolution about the $\overrightarrow{p1p2}$ vector.

A sketch of the definition is given in the figure below.

Assertions:

- $0 \leq$ *internal_diameter* $<$ *external_diameter*
- $-360° <$ *start_angle* $\leq 360°$
- $-360° <$ *end_angle* $\leq 360°$
- $0° \leq$ *end_angle* $-$ *start_angle* $\leq 360°$

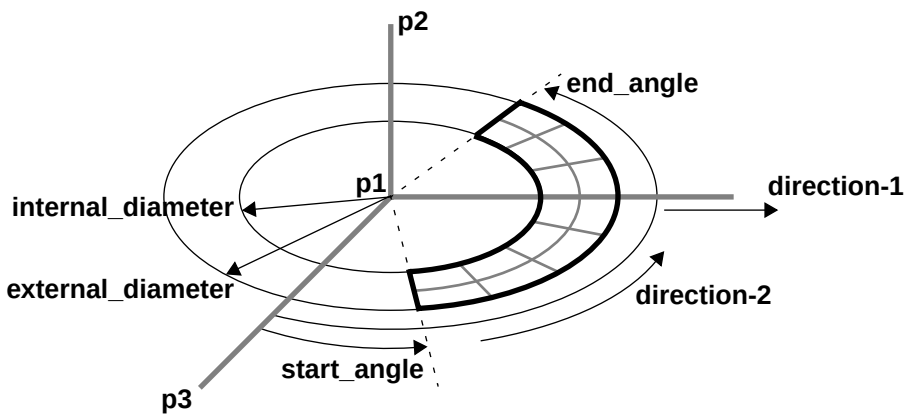


Figure 4–10 - Sketch of a disc (actually a sector of a ring)

4.2.9.7.6 cylinder

A **cylinder** is a **primitive_bounded_surface**, which defines a cylinder or cylinder-segment surface.

NOTE 61 – The attributes and assertions are in addition to those inherited from **primitive_bounded_surface**.

Attributes:

- *p1:* reference to a **geometric_construction_point**
  centre of the base of the cylinder and first point of the axis of revolution
- *p2:* reference to a **geometric_construction_point**
  centre of the top of the cylinder and second point of the axis of revolution
- *diameter:* length_measure
- *p3:* reference to a **geometric_construction_point** to define the p1p2p3 reference plane for the *start_angle* and *end_angle*
- *start_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the start of the segment
- *end_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the end of the segment

The positive normal on the surface is defined outwards from the cylinder or cylinder-segment.

The first parametric direction (direction-1) is defined along the $\overrightarrow{p1p2}$ direction, the second parametric direction (direction-2) is defined along the positive direction of revolution about the $\overrightarrow{p1p2}$ vector.

A sketch of the definition is given in the figure below.

Assertions:

- *diameter* > 0
- -360° < *start_angle* ≤ 360°
- -360° < *end_angle* ≤ 360°
- 0° ≤ *end_angle* – *start_angle* ≤ 360°

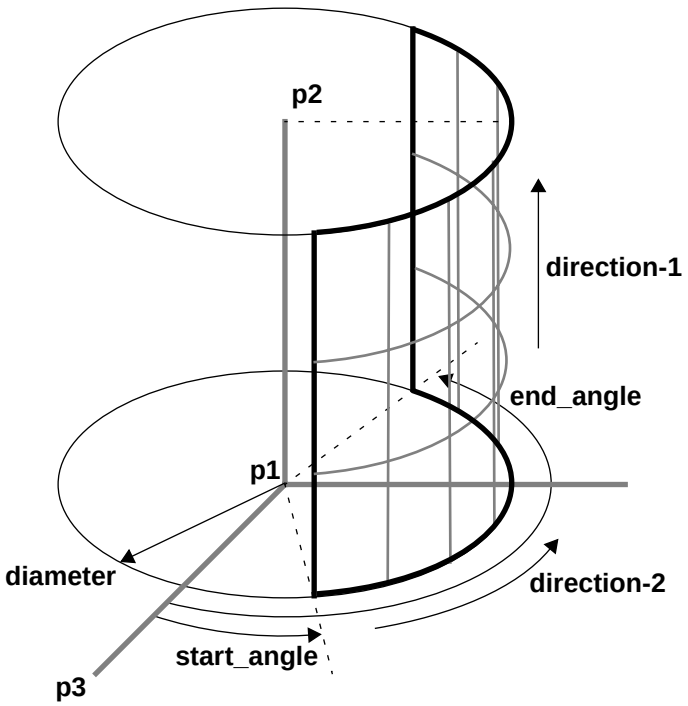


Figure 4–11 - Sketch of a cylinder (actually a cylinder-segment)

4.2.9.7.7 cone

A **cone** is a **primitive_bounded_surface**, which defines a cone or cone-segment surface.

NOTE 62 – The attributes and assertions are in addition to those inherited from **primitive_bounded_surface**.

Attributes:

- *p1:* reference to a **geometric_construction_point**
  centre of the base of the cone and first point of the axis of revolution
- *p2:* reference to a **geometric_construction_point**
  centre of the apex of the cone and second point of the axis of revolution
- *base_diameter:* length_measure
- *apex_diameter:* length_measure
- *p3:* reference to a **geometric_construction_point** to define the p1p2p3 reference plane for the *start_angle* and *end_angle*
- *start_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the start of the segment
- *end_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the end of the segment

The positive normal on the surface is defined outwards from the cylinder or cylinder-segment.

The first parametric direction (direction-1) is defined along the $\overrightarrow{p1p2}$ direction, the second parametric direction (direction-2) is defined along the positive direction of revolution about the $\overrightarrow{p1p2}$ vector.

A sketch of the definition is given in the figure below.

Assertions:

- *base_diameter* > 0
- *apex_diameter* ≥ 0
- -360° < *start_angle* ≤ 360°
- -360° < *end_angle* ≤ 360°
- 0° ≤ *end_angle* – *start_angle* ≤ 360°

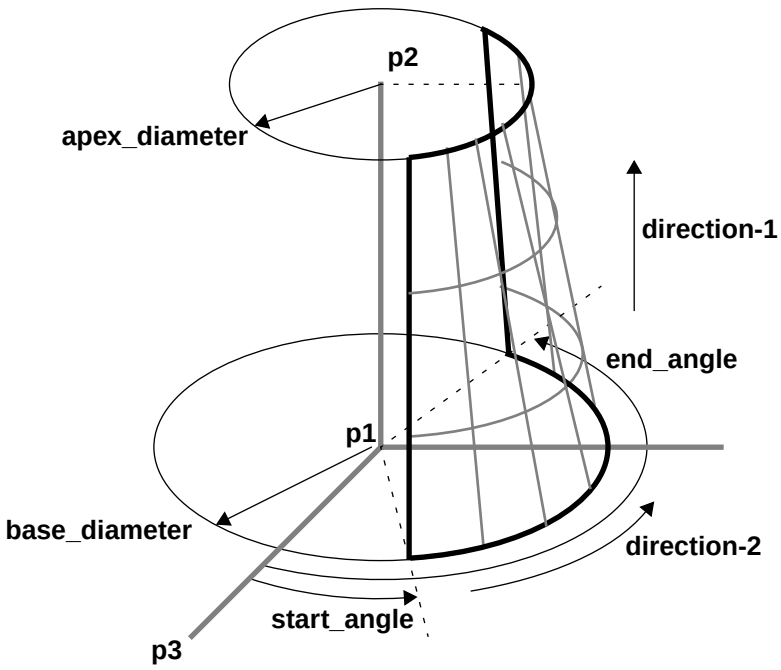


Figure 4–12 - Sketch of a cone (actually a truncated cone-segment)

4.2.9.7.8 sphere

A **sphere** is a **primitive_bounded_surface**, which defines a complete or partial spherical surface.

NOTE 63 – The attributes and assertions are in addition to those inherited from **primitive_bounded_surface**.

Attributes:

- *p1:* reference to a **geometric_construction_point**
  centre of the sphere and first point of the axis of revolution
- *p2:* reference to a **geometric_construction_point**
  second point of the axis of revolution
- *diameter:* **length_measure** – diameter of the sphere
- *base_truncation:* **length_measure** along the $\overrightarrow{p1p2}$ axis defining the base of a truncated sphere – is set to *–diameter / 2* for a complete sphere
- *apex_truncation:* **length_measure** along the $\overrightarrow{p1p2}$ axis defining the apex of a truncated sphere – is set to *diameter / 2* for a complete sphere
- *p3:* reference to a **geometric_construction_point** to define the p1p2p3 reference plane for the *start_angle* and *end_angle*
- *start_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the start of the segment
- *end_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the end of the segment

The positive normal on the surface is defined outwards from the sphere or sphere-segment.

The first parametric direction (direction-1) is defined along the $\overrightarrow{p1p2}$ direction, the second parametric direction (direction-2) is defined along the positive direction of revolution about the $\overrightarrow{p1p2}$ vector.

A sketch of the definition is given in the figure below.

Assertions:

- $-diameter/2 \leq base_truncation < apex_truncation \leq diameter/2$
- $-360° < start_angle \leq 360°$
- $-360° < end_angle \leq 360°$
- $0° \leq end_angle - start_angle \leq 360°$

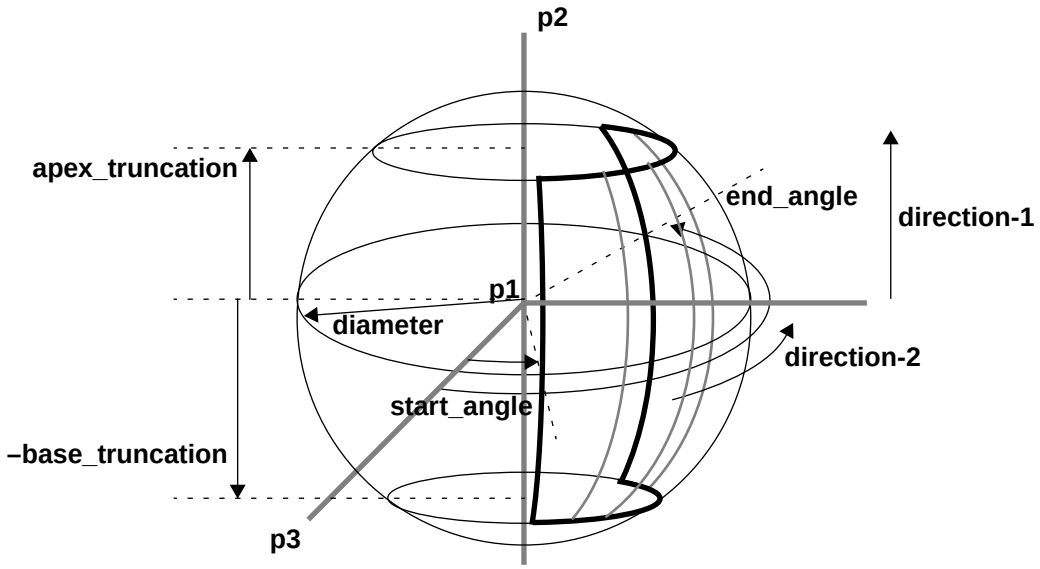


Figure 4–13 - Sketch of a sphere (actually a truncated sphere-segment)

4.2.9.7.9 paraboloid

A **paraboloid** is a **primitive_bounded_surface**, which defines a complete or partial paraboloid surface.

NOTE 64 – The attributes and assertions are in addition to those inherited from **primitive_bounded_surface**.

Attributes:

- *p1:* reference to a **geometric_construction_point**
  apex of the paraboloid and first point of the axis of revolution
- *p2:* reference to a **geometric_construction_point**
  centre of the base of the paraboloid and second point of the axis of revolution
- *diameter:* **length_measure**, which is the diameter of the base of the paraboloid
- *base_truncation:* **length_measure** from *p1* along the $\overrightarrow{\mathrm{p1p2}}$ axis defining the base of a truncated paraboloid – set to length of $\overrightarrow{\mathrm{p1p2}}$ when no truncation is desired
- *apex_truncation:* **length_measure** from *p1* along the $\overrightarrow{\mathrm{p1p2}}$ axis defining the apex of a truncated paraboloid – set to 0 when no truncation is desired
- *p3:* reference to a **geometric_construction_point** to define the p1p2p3 reference plane for the *start_angle* and *end_angle*
- *start_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the start of the segment
- *end_angle:* **plane_angle_measure** relative to p1p2p3 plane, which defines the end of the segment

The positive normal on the surface is defined outwards from the paraboloid or paraboloid-segment.

The first parametric direction (direction-1) is defined along the $\overrightarrow{\mathrm{p1p2}}$ direction, the second parametric direction (direction-2) is defined along the positive direction of revolution about the $\overrightarrow{\mathrm{p1p2}}$ vector.

A sketch of the definition is given in the figure below.

Assertions:

- $diameter > 0$
- $0 \leq apex_truncation < base_truncation \leq \text{length } \overrightarrow{\mathrm{p1p2}}$
- $-360° < start_angle \leq 360°$
- $-360° < end_angle \leq 360°$
- $0° \leq end_angle - start_angle \leq 360°$

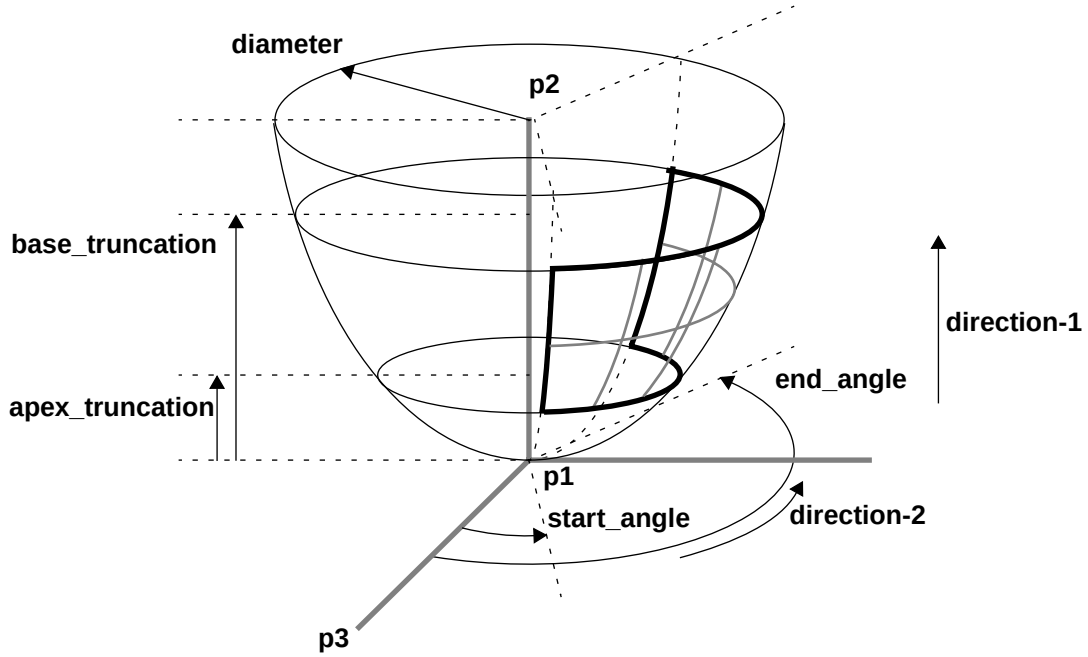


Figure 4–14 - Sketch of a paraboloid (actually a truncated paraboloid-segment)

4.2.9.8 Compound bounded surfaces

4.2.9.8.1 compound_surface_shell

A **compound_surface_shell** is a special kind of a bounded surface. It is a named grouping of a number of connected **primitive_bounded_surface**s, which together form an open or a closed shell.

Attributes:

- *id:* identifier string
- *name:* label string
- *surfaces:* list of two or more references to **primitive_bounded_surface**s and/or **compound_surface_shell**s

Assertions:

- Any **primitive_bounded_surface** in the sur*faces* list shall be connected to at least one other **primitive_bounded_surface** in the *surfaces* list through a common edge and/or vertex.

EXAMPLE 19 – In the figure below two sketches are given: for a box with 6 faces and for a tank.
The *surfaces* list of the box contains 6 references to **rectangle**s;
The *surfaces* list of the tank contains 1 reference to a **cylinder** and 2 references to semi-**sphere**s.

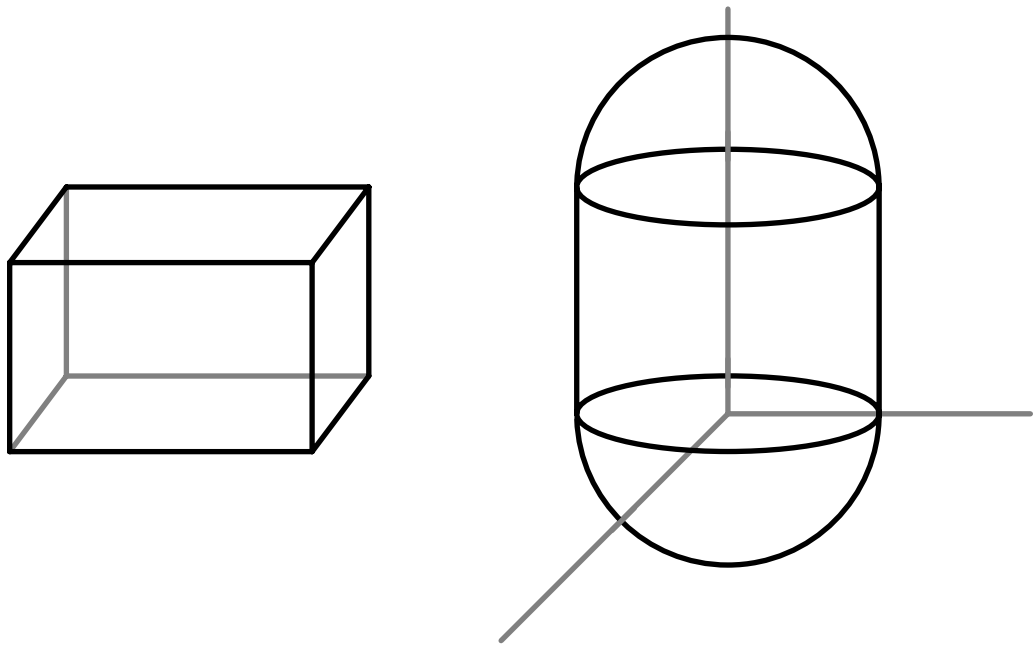


Figure 4–15 - Sketches of two examples of a compound_surface_shell: a *box with 6 faces* and a *tank*

4.2.9.9 Auxiliary boolean construction surfaces

4.2.9.9.1 boolean_construction_surface

A **boolean_construction_surface** is a half space limited by an unbounded surface. It is a generic object which specifies the attributes that are common to all auxiliary boolean construction surfaces.

Attributes:

- *id:* identifier string
- *sense:* one item from the enumeration {positive, negative}, which specifies the side of the surface to be used in boolean construction operations. If *sense* equals positive then the half space is the portion of space the normal of the underlying surface in any point points away from. If *sense* equals negative, then the half space is the region in space towards where the normal of the underlying surface points.
- *placement:* reference to an **axis_placement,** which defines the location and orientation for the surface with respect to the coordinate system of the **geometric_model** in which the surface is defined

4.2.9.9.2 half_space_plane

A **half_space_plane** is a type of **boolean_construction_surface**, which specifies a half space by defining an infinite plane. The half space comprises all points that lie on one side of the plane.

The *location* of the *placement* is a point on the plane;

The normal on the plane is defined by the *z_axis_direction* of the *placement.*

Attributes:

- None

Assertions:

- None

Notes:

NOTE 65 – The attributes and assertions are in addition to those inherited from **boolean_ construction_surface**.

4.2.9.9.3 half_space_cylinder

A **half_space_cylinder** is a type of **boolean_construction_surface**, which specifies a half space by defining an infinite cylindrical surface.

The *location* of the *placement* is a point on the cylinder axis;

The *z_axis_direction* of the *placement* defines the orientation of the cylinder axis;

The normal points outwards of the cylindrical surface.

Attributes:

- *radius*: **length_measure**, specifying the radius of the cylindrical surface

Assertions:

- None

Notes:

NOTE 66 – The attributes and assertions are in addition to those inherited from **boolean_ construction_surface**.

4.2.9.9.4 generalized_half_space

A **generalized_half_space** is a half space defined by the underlying surface of a **primitive_bounded_ surface**. The half space comprises all points that lie on the side of the surface specified by the attribute *sense*.

If *sense* equals positive then the side is the portion of space the normal of the underlying surface in any point points away from. If *sense* equals negative, then the half space is the region in space towards where the normal of the underlying surface points.

Attributes:

- *base_surface*: specifies the **primitive_bounded_surface**
- *sense:* one item from the enumeration {positive, negative}, which specifies the side of the surface to be used in boolean construction operations.

Assertions:

- None

4.2.9.10 Boolean result surfaces

A boolean result surface is a complex surface that results from the execution of one of the boolean construction operations on two operand surfaces. The possible boolean construction operations are:

- union – the resulting surface comprises the combination of the two operand surfaces;
- intersection – the resulting surface comprises the common parts of the two operand surfaces;
- difference – the resulting surface comprises the first operand surface minus its intersection with the boolean construction solid defined by the second operand surface.

Assertions:

In this protocol only the difference operation shall be supported.

NOTE 67 – The difference operation is also often called 'cut' operation.

4.2.9.10.1 boolean_result_surface

A **boolean_result_surface** is a general object that captures the data that is common to all boolean result surfaces, which are bounded surfaces.

Attributes:

- *id*: identifier string
- *placement*: OPTIONAL reference to an **axis_placement**, that specifies a displacement to be applied to the result of the boolean operation. The displacement is defined as the transformation which maps the coordinate system of the geometric model containing the **boolean_result_surface** onto *placement*.

4.2.9.10.2 boolean_difference_surface

A **boolean_difference_surface** is a **boolean_result_surface** that is formed by a boolean difference operation between one surface and another surface.

Attributes:

- *first_operand*: reference to a **primitive_bounded_surface**, a **compound_bounded_surface** or a **boolean_difference_surface**
- *second_operand*: reference to a **boolean_construction_surface** or a **generalized_half_space.**

Assertions:

- None

Notes:

NOTE 68 – The attributes and assertions are in addition to those inherited from **boolean_result_surface**.

NOTE 69 – The boolean difference operation is defined as follows:
A boolean construction solid is defined through the *second_operand* and its *sense* attribute;
The intersection is computed between the *first_operand* and the boolean construction solid;
The resulting **boolean_difference_surface** is the *first_operand* surface minus the part(s) that intersect with the boolean construction solid;
The *placement* shall be effectuated on the evaluated surface resulting from the boolean operation.

Examples:

EXAMPLE 20 – A typical example is a hole cut in a **rectangle** with a **cylinder**

4.2.10 kinematic_model UoF

NOTE 70 – The terminology and concepts used in this UoF are as close as possible to [STEP-105].

4.2.10.1 kinematic_model

A **kinematic_model** is a model that defines the kinematic structure and capabilities of a **product** represented by a **geometric_model**.

The basic entities in a kinematic model are the **kinematic_link**, the **kinematic_joint** and the **kinematic_pair**. The **kinematic_link** specifies the existence of a rigid body, while the **kinematic_joint** specifies the possibility of relative motion between two rigid bodies. Together a set of **kinematic_**

links and **kinematic_joint**s establish the topology of a kinematic chain. A **kinematic_pair** specifies for a particular **kinematic_joint** how its motion is constrained, e.g. that it has one rotational degree of freedom and that it has given lower and upper limits on its allowable angle of rotation.

In this protocol a **kinematic_model** can not exist without an associated **geometric_model**. The topology of the **kinematic_model** is limited to open kinematic chains – i.e. its graph representation is a single oriented tree, which starts at some kinematic base and progresses through a number of kinematic links and joints to the last link(s) in the chain. In other words, no kinematic networks or loops are allowed. Furthermore the structure of the kinematic chain shall agree with the model/submodel tree of the associated **geometric_model**, which means that:

- the *shape_representation*s of all **kinematic_link**s in one **kinematic_model** shall refer to the root **geometric_model** or to one of its **geometric_submodel_usage**s
- **kinematic_joint**s are only possible between **kinematic_link**s whose *shape_representation*s are related via **geometric_submodel_usage**s – within one geometric model/submodel tree.

Attributes:

- *id:* identifier
- *name:* label
- *description:* text
- *joints*: set of one or more references to **kinematic_joint**s
- *base*: reference to a **kinematic_link** which specifies the kinematic base (see subclause 3.5.2 on page 3-3) of the **kinematic_model** – this is a derived attribute which references the **kinematic_link** whose *shape_representation* references the root **geometric_model**

Assertions:

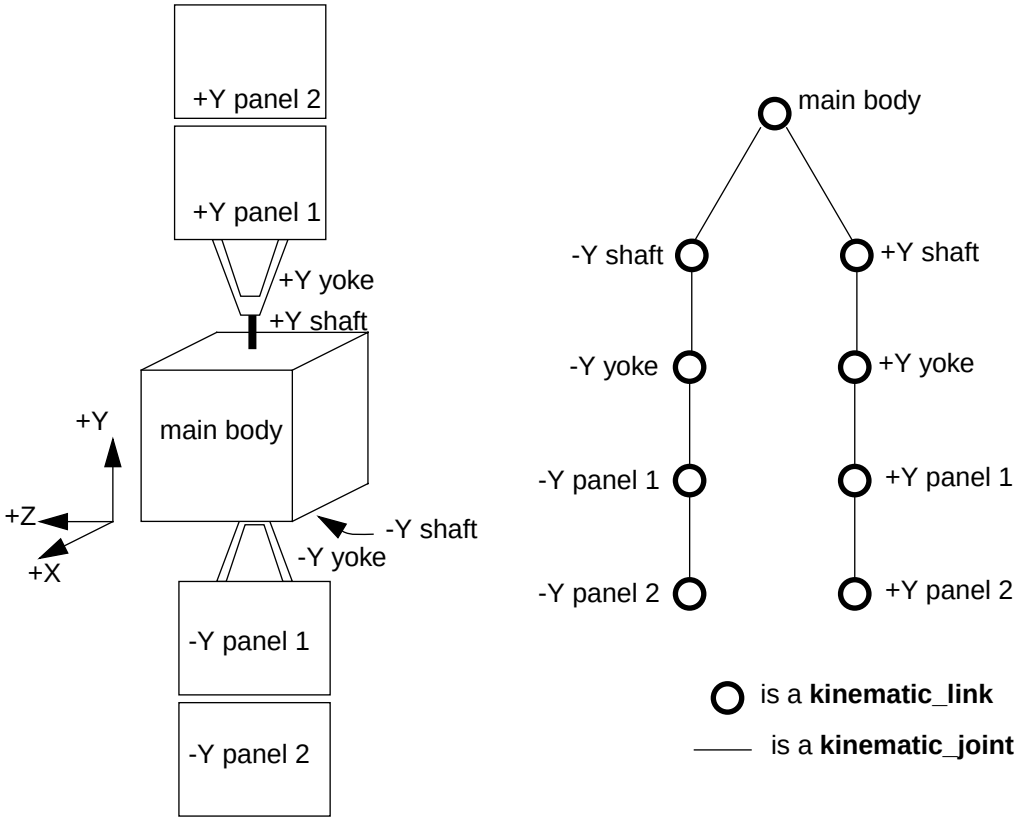
- The *shape_representation*s of all **kinematic_link**s referred to in *joints* shall belong to one geometric model/submodel tree (as specified by a number of **geometric_model**s and **geometric_submodel_usage**s)
- Each *joints* shall refer to a unique pair of **kinematic_link**s – i.e. no more than one **kinematic_joint** can exist between two **kinematic_link**s
- In the set of all **kinematic_link**s associated to the *joints*, one and only one **kinematic_link** shall reference through its *shape_representation* attribute the root **geometric_model** of the geometric model/submodel tree

Notes:

WORKING NOTE 6 – This **kinematic_model** maps well to the [STEP-105] entities **mechanism** and **kinematic_structure**.

Examples:

EXAMPLE 21 – In the figure below the main concepts for a kinematic model are illustrated for a typical spacecraft configuration.



(a) Schematic shape model of a typical communications satellite with two fully deployed solar array wings

(b) The corresponding **kinematic_model** presented as a topological graph

Figure 4–16 - Illustration of a kinematic model for a typical spacecraft configuration

4.2.10.2 kinematic_link

A **kinematic_link** is a specification of the topologic aspects of a rigid body part in a kinematic chain, whose motion is constrained by one or more **kinematic_joint**s. The shape representation of a **kinematic_link** shall be defined by a **geometric_model** or a **geometric_submodel_usage**.

Attributes:

- *shape_representation*: reference to a **geometric_model**, which defines only the shape of the link, not its position nor its orientation
- *associated_product*: reference to a **product_definition** or a **product_next_higher_assembly_usage_relationship**.
  If a **product_definition** is referenced, it shall be the root of the product assenbly tree – i.e. it shall not be referred to in any **product_next_higher_assembly_usage_relationship** as a *constituent*.
  Else if a **product_next_higher_assembly_usage_relationship** is referenced, the link is associated with the *constituent*.

Assertions:

- If the *shape_representation* references a **geometric_model** it shall be the root **geometric_model**

Notes:

NOTE 71 – In this protocol all **geometric_model**s (and therefore all **geometric_submodel_usage**s) are considered to represent rigid bodies.

4.2.10.3 kinematic_joint

A **kinematic_joint** is a specification of the possibility of kinematic articulation between two and only two **kinematic_link**s.

Attributes:

- *first_link*: reference to a **kinematic_link**
- *second_link*: reference to a **kinematic_link**

Assertions:

- The *first_link* and the *second_link* shall not refer to the same **kinematic_link**
- The *associated_product* of the *first_link* shall refer to a **product_definition** which is the top **product_definition** in a product assembly tree or to a **product_next_higher_assembly_usage_relationship** which is part of the product assembly tree.
- The *associated_product* of the *second_link* shall refer to a **product_next_higher_assembly_usage_relationship** which is part of the product assembly tree of which the *associated_product* of the *first_link* is also a part.

4.2.10.4 kinematic_pair

A **kinematic_pair** is a specification of the kinematic restrictions of a **kinematic_joint**. It is a generic object that will not be instantiated itself, but provides a common mechanism for reference.

Attributes:

- *id:* identifier string
- *joint*: reference to a **kinematic_joint**

Assertions:

- There shall be a maximum of one **kinematic_pair** for one **kinematic_joint**

Notes:

NOTE 72 – A **kinematic_pair** overrides any static positioning and orientation that may have been specified – through the *axis_placement* attribute of a **geometric_submodel_usage** – for a **geometric_model** that is associated with a **kinematic_link**.

4.2.10.5 kinematic_pair_value

A **kinematic_pair_value** is a specification of a kinematic configuration of a **kinematic_pair**. It is a generic object that will not be instantiated itself, but provides a common mechanism for reference.

Attributes:

- *applies_to_pair:* reference to a **kinematic_pair**

4.2.10.6 revolute_kinematic_pair

A **revolute_kinematic_pair** is a **kinematic_pair** with one rotational degree of freedom.

Attributes:

- *rotation_axis*: reference to an **axis_placement**, specifying the axis of rotation for the *second_link* in the coordinate system of the *first_link* of the *joint*. The axis of rotation is along the *z_axis_direction* of the **axis_placement** and its *x_axis_direction* defines the reference plane for the 0° rotation angle.
- *lower_limit_actual_rotation*: OPTIONAL **plane_angle_measure**, which specifies the smallest allowable rotation angle
- *upper_limit_actual_rotation*: OPTIONAL **plane_angle_measure**, which specifies the largest allowable rotation angle

Assertions:

- *lower_limit_actual_rotation* < *upper_limit_actual_rotation*, when both are defined

Notes:

NOTE 73 – The attributes and assertions are in addition to those inherited from **kinematic_pair**.

4.2.10.7 explicit_revolute_kinematic_pair_value

An **explicit_revolute_kinematic_pair_value** is a **kinematic_pair_value** that specifies explicitly the actual rotation angle of a **revolute_kinematic_pair**.

Attributes:

- *actual_rotation*: **plane_angle_measure**, specifying the actual angle

Assertions:

- *applies_to_pair* shall reference a **revolute_kinematic_pair**
- If *upper_limit_actual_rotation* of *applies_to_pair* is defined, then *actual_rotation* ≤ *upper_limit_actual_rotation*
- If *lower_limit_actual_rotation* of *applies_to_pair* is defined, then *actual_rotation* ≥ *lower_limit_actual_rotation*

Notes:

NOTE 74 – The attributes and assertions are in addition to those inherited from **kinematic_pair_value**.

4.2.10.8 implicit_revolute_kinematic_pair_value

An **implicit_revolute_kinematic_pair_value** is a **kinematic_pair_value** that specifies the actual angle of a **revolute_kinematic_pair** through the specification of a pointing direction for a given axis of the*second_link* of the **kinematic_joint** that is associated with the **implicit_revolute_kinematic_pair_value**.

Attributes:

- *pointing*: reference to a **pointing_in_space**, which specifies a direction in space that a given axis (the *pointer* of the **pointing_in_space**) of the *second_link* shall be pointed to
- *may_modify_pointing_of_first_link*: a boolean flag, indicating whether or not the pointing of the rigid body represented by the *first_link* of the **kinematic_joint** (that is associated with the **implicit_revolute_kinematic_pair_value**) may be modified within its kinematic constraints in order to achieve the best possible pointing for the *second_link*

Assertions:

- *applies_to_pair* shall reference a **revolute_kinematic_pair**
- The direction of the *submodel_pointer* shall not coincide with the direction of the *rotation_axis* of the *applies_to_pair*

Notes:

NOTE 75 – The attributes and assertions are in addition to those inherited from **kinematic_pair_value**.

NOTE 76 – The receiving software is responsible for computing the rotation angle at which a best *pointing* of the *second_link* is achieved, while meeting the restrictions defined for the *pair*.

NOTE 77 – Setting *may_modify_pointing_of_first_link* true only makes sense when the *first_link* itself has at least one rotational degree of freedom.

WORKING NOTE 7 – *may_modify_pointing_of_first_link* corresponds to the PARENT=YES or NO attribute of the POINT statement in ESABASE's $KINEMATIC section, see [ESABASE-RM].

Examples:

EXAMPLE 22 – A typical example in space applications is the kinematic system of a three-axes stabilised earth-pointing spacecraft and its rotating solar panels that track the sun (cf. a telecommunications satellite in a geostationary orbit).

EXAMPLE 23 – An example of the application of *may_modify_pointing_of_first_link* (set TRUE) is a spacecraft with a telescope and solar panels which can rotate perpendicular to the telescope's main axis (cf. the Hubble Space Telescope). The telescope would be pointed towards a star of interest, leaving free one rotational degree of freedom of the main body. An optimal combination of rotations about the telescope's main axis and about the solar panels' rotation axis can then be computed to receive maximum solar input on the panels.

EXAMPLE 24 – Another typical example in space applications is the kinematic system of a low earth orbit spacecraft and its data relay antenna, which is suspended in a gimbal. The antenna tracks a geostationary relay satellite. The gimbal mechanism would be modelled by two **implicit_revolute_kinematic_motion**s and an intermediate submodel (through a **geometric_submodel_usage**).

4.2.10.9 prismatic_kinematic_pair

A **prismatic_kinematic_pair** is a **kinematic_pair** with one translational (or: sliding) degree of freedom.

Attributes:

- *sliding_axis*: reference to an **axis_placement**, specifying the axis along which translation is possible for the *second_link* in the coordinate system of the *first_link* of the *joint*. The positive translation direction is along the *placement*'s *z_axis_direction* and its XY-plane defines the zero sliding distance.
- *lower_limit_actual_translation*: OPTIONAL **length_measure**, which specifies the smallest allowable sliding distance, which may have a negative value
- *upper_limit_actual_translation*: OPTIONAL **length_measure**, which specifies the largest allowable sliding distance, which may have a negative value

Assertions:

- *lower_limit_actual_translation* < *upper_limit_actual_translation*, when both are defined

Notes:

NOTE 78 – The attributes and assertions are in addition to those inherited from **kinematic_pair**.

4.2.10.10 prismatic_kinematic_pair_value

An **prismatic_kinematic_pair_value** is a **kinematic_pair_value** specifying explicitly the sliding distance for a **prismatic_kinematic_pair_value**.

Attributes:

- *actual_translation*: **length_measure**, specifying the actual translation along the *sliding_axis* of the **prismatic_kinematic_pair** referenced in *applies_to_pair*

Assertions:

- *applies_to_pair* shall reference a **prismatic_kinematic_pair**
- If *upper_limit_actual_translation* of *applies_to_pair* is defined, then *actual_translation* ≤ *upper_limit_actual_translation*
- If *lower_limit_actual_translation* of *applies_to_pair* is defined, then *actual_translation* ≥ *lower_limit_actual_translation*

Notes:

NOTE 79 – The attributes and assertions are in addition to those inherited from **kinematic_pair_value**.

4.2.11 thermal_radiative_model UoF

4.2.11.1 basic_thermal_radiative_model

A **basic_thermal_radiative_model** is a type of **network_model**, specifying a model representation with oriented sets of thermal-radiative faces for the purpose of thermal-radiative analysis. The faces are associated with bounded surfaces in a **geometric_model**.

Attributes:

- None

Assertions:

- All *constituents* shall be **thermal_radiative_face_set**s

Notes:

NOTE 80 – The attributes and assertions are in addition to those inherited from **network_model**.

4.2.11.2 thermal_radiative_face_set

A **thermal_radiative_face_set** is a type of **network_node**, specifying a connected set of thermal-radiative faces defined on one side of a bounded surface.

Attributes:

- *surface*: reference to a **primitive_bounded_surface**
- *active_side*: one item from the enumeration {none, positive, negative}
- *assigned_material*: OPTIONAL reference to a **material** which is assigned to the **thermal_radiative_face_set** for thermal-radiative analysis purposes
- *notional_thickness*: OPTIONAL **length_measure**, which specifies a notional thickness that may be used to derive a 'pseudo-solid' object for analysis purposes. The referenced *surface* acts as a neutral fibre (or: neutroid) surface for the 'pseudo-solid'. The boundaries of the 'pseudo-solid' are defined as follows:
  - one bounded surface at the positive side of the nominal surface, at an offset of half the *notional_thickness* in the direction of the local surface-normal;
  - one bounded surface at the negative side, similarly offset as above;
  - the surfaces which are formed by the edges of the two offset surfaces and – where applicable – the edges between the corresponding corner vertices of the offset surfaces at the positive and negative sides.

Notes:

NOTE 81 – The attributes and assertions are in addition to those inherited from **network_node**.

NOTE 82 – The positive side of each kind of **primitive_bounded_surface** is specified in its definition in subclause 4.2.9.7 on page 4-25.

NOTE 83 – *active_side* equals "none" denotes that a **thermal_radiative_face_set** does not emit (or reflect) thermal radiation, but it is present as a shade in the thermal-radiative model, with an artificial absorptance of 1 for all thermal radiation.

NOTE 84 – There are two ways in which material properties can be associated with a **thermal_radiative_face_set**.

- The first way is through a **material_product_relation** and a **model_represents_product_relationship** for the **thermal_radiative_face_set** of interest. The two relationships establish a material with associated properties for the **thermal_radiative_face_set**.
- The second way is through the *assigned_material* attribute, which overrides any possible existing association described as the first way. This second possibility is defined to allow for the cases where the product definition breakdown is not (yet) sufficiently detailed.

It is expected that for such an associated *material* at least the first two – but often all – of the following thermo-optical properties are specified:

- solar absorptance
- infra-red emittance
- solar and infra-red transmittance
- solar and infra-red diffuse_transmittance
- specular_reflection_ratio
- diffuse_specularity_ratio

4.2.11.3 meshed_thermal_radiative_model

A **meshed_thermal_radiative_model** is a type of **network_model**, which specifies the thermal meshes for the **thermal_radiative_face_set**s of a **basic_thermal_radiative_model**.

Effectively, it contains the definition of thermal-radiative faces and their aggregation mapping to **thermal_lumped_parameter_node**s.

Attributes:

- None

Assertions:

- All *constituents* shall be **thermal_radiative_meshing**s

- All **thermal_radiative_face_set** instances referenced in the *face_set* of the *constituents* (which are **thermal_radiative_meshing**s) shall belong to one **basic_thermal_radiative_model** instance.

Notes:

NOTE 85 – The attributes and assertions are in addition to those inherited from **network_model**.

4.2.11.4 thermal_radiative_meshing

A **thermal_radiative_meshing** is a type of **network_node** which specifies the meshing of a **thermal_radiative_face_set** into thermal-radiative faces and assigns thermal node identifiers to the thermal-radiative faces. Through this mechanism an aggregation mapping from thermal-radiative faces to thermal lumped parameter nodes (via the *id* of **thermal_lumped_parameter_node**s) is specified. It is a generic object that will not be instantiated itself, but defines the attributes and assertions common to all **thermal_radiative_meshing**s and provides a common mechanism for reference.

Attributes:

- *face_set:* reference to a **thermal_radiative_face_set**
- *meshings_1*: integer ≥ 1, the number of mesh divisions along the first iso-parametric direction
- *meshings_2*: integer ≥ 1, the number of mesh divisions along the second iso-parametric direction
- *numbering_option*: one item from the enumeration {DIRECTION_1_FIRST, DIRECTION_2_FIRST}
- *grid_spacings*: OPTIONAL list of coordinates in iso-parametric space, specifying the (possibly non-uniform) spacing of the meshing grid in both iso-parametric directions. The list is formed by a sequence of *meshings_1*+1 strictly increasing real numbers between 0 and 1 to specify the grid spacing in the first iso-parametric direction, followed by a sequence of *meshings_2*+1 strictly increasing real numbers between 0 and 1 to specify the grid spacing in the second iso-parametric direction.

Assertions:

- If *grid_spacings* is defined,
  the list shall have (*meshings_1*+1+*meshings_2*+1) elements,
  all elements shall have a value ≥ 0 and ≤ 1,
  element [1] shall be 0,
  element[*meshings_1*] shall be 1,
  element[*meshings_1*+1] shall be 0,
  element[*meshings_1*+1+*meshings_2*+1] shall be 1,
  elements [1] to [*meshings_1*] shall have strictly increasing values,
  elements [*meshings_1*+1] to [*meshings_1*+1+*meshings_2*+1] shall have strictly increasing values

Notes:

NOTE 86 – The attributes and assertions are in addition to those inherited from **network_node**.

NOTE 87 – Effectively *meshings_1* • *meshings_2* thermal-radiative faces are designated for *face_set*.

NOTE 88 – For *numbering_option*, DIRECTION_1_FIRST means first specify the id's of thermal nodes along the first iso-parametric direction and then along the second iso-parametric direction; DIRECTION_2_FIRST means first increment the id's of thermal nodes along the second iso-parametric direction and then along the first iso-parametric direction

NOTE 89 – If *grid_spacings* is left undefined, uniform meshing is implied, in the way as illustrated in the typical meshings drawn in the figures in subclause 4.2.9.7 "Primitive bounded surfaces". A uniform meshing is constructed by dividing the edge(s) in the first iso-parametric direction in *meshings_1* equidistant divisions and the edge(s) in the second iso-parametric direction in *meshings_2* equidistant divisions, thereby defining *meshings_1* • *meshings_2* faces which are topologically triangular or rectangular (i.e. they correspond to a triangle or rectangle in parametric space – for a more detailed description, see also [STEP-42], subclause 4.4.70 on rectangular_composite_surface).

4.2.11.5 thermal_radiative_meshing_with_incremental_node_list

A **thermal_radiative_meshing_with_incremental_node_list** is a **thermal_radiative_meshing**, specifying a list of target thermal lumped parameter node-id's by a first node-id and an incremental value. The target list specifies the thermal lumped parameter node-id's to which results for the faces in the *face_set* are to be aggregated.

Attributes:

- *first_thermal_node_id*: OPTIONAL positive integer specifying the identifier for the thermal lumped parameter node to which thermal-radiative analysis results for the first face in the *face_set* are to be mapped
- *thermal_node_id_increment*: positive integer specifying the increment that shall be used to construct a list of identifiers for thermal lumped parameter nodes for all (*meshings_1* • *meshings_2*) faces in the *face_set*

Notes:

NOTE 90 – The attributes and assertions are in addition to those inherited from **thermal_radiative_meshing**.

NOTE 91 – Typically thermal lumped parameter node-id's fall in the range 1 to 99999 inclusive.

NOTE 92 – If the *first_thermal_node_id* is left undefined, then the actually assigned node identifiers for the faces in the *face_set* is determined by the receiving postprocessor

4.2.11.6 thermal_radiative_meshing_with_full_node_list

A **thermal_radiative_meshing_with_full_node_list** is a **thermal_radiative_meshing**, specifying a list of target thermal lumped parameter node-id's by an explicit, full list of node-id's for all meshed faces in a **thermal_radiative_face_set**. The target list specifies the thermal lumped parameter node-id's to which results for the faces in the *face_set* are to be aggregated.

Attributes:

- *thermal_node_ids*: list of positive integers denoting the thermal lumped parameter node-id's to which thermal-radiative analysis results for the faces in the *face_set* are to be mapped. The same node-id may be specified more than once.

Assertions:

- The *thermal_node_ids* list must have *meshings_1* • *meshings_2* elements, i.e. one identifier for each meshed face in the *face_set*

Notes:

NOTE 93 – The attributes and assertions are in addition to those inherited from **thermal_radiative_meshing**.

4.2.11.7 enclosure

An **enclosure** is a named grouping of **thermal_radiative_face_set** objects for the purpose of efficient thermal-radiative computations. The thermal-radiative faces in one enclosure can not exchange heat through thermal radiation with faces in another enclosure.

Attributes:

- *of_model*: reference to a **basic_thermal_radiative_model**, for which the enclosure is specified
- *id*: identifier string
- *name*: label string
- *face_sets*: list of references to **thermal_radiative_face_set** objects belonging to the enclosure

Assertions:

- The combination *of_model* and *id* shall be unique in the exchange dataset
- All *face_sets* belong to the **basic_thermal_radiative_model** specified in *of_model* and its submodels

4.2.12 thermal_lumped_parameter_model UoF

WORKING NOTE 8 – This item is identified for a future version and not in scope for the AIM.

WORKING NOTE 9 – A **thermal_lumped_parameter_model** can be fully represented in a [STEP-NRF-AP] **network_model**, containing the following constituents:
- **network_node**s with *class* "`thermal_lumped_parameter_node`";
- **network_node_relationship**s with *class* "`radiative_exchange_factor`", "`thermal_radiative_coupling`", "`thermal_conductor`" or "`mass_flow_link`".

4.2.13 thermal_finite_element_model UoF

WORKING NOTE 10 – This item is identified for a future version and not in scope for the AIM.

WORKING NOTE 11 – A **thermal_finite_element_model** can be represented by a [STEP-NRF-AP] **network_model**.

4.2.14 visual_presentation UoF

4.2.14.1 colour_rgb

A **colour_rgb** is a specification of a colour definition in terms of its red, green and blue (RGB) primary constituents, as defined in [STEP-46]. In this protocol it is only used to convey colour definitions for presentation purposes, i.e. colour as an informational tag, not as a property of a product part.

Attributes:
- *red:* fraction red constituent
- *green:* fraction green constituent
- *blue:* fraction blue constituent

Assertions:
- $0 \leq red \leq 1$
- $0 \leq green \leq 1$
- $0 \leq blue \leq 1$

4.2.15 material_properties UoF

4.2.15.1 material

A **material** is a simple specification of a material for use in analysis models. A specific surface treatment may be specified.

Attributes:
- *id:* identifier string
- *name*: label string, denoting the generic name of the material – e.g. "Aluminium"
- *designation*: OPTIONAL label string, denoting the material designation – e.g. "5056"
- *surface_treatment*: OPTIONAL label string, denoting possible treatment of the material's surface – e.g. "polished"
- *description*: OPTIONAL text, giving a description of the material

Assertions:
- The *id* shall be unique in the exchange dataset

Examples:

EXAMPLE 25 – Machined aluminium 3036 and polished aluminium 5056 are **material**s, with different *surface_treatment* and *designation* .

4.2.15.2 product_material_relationship

A **product_material_relationship** is a specification of the materials that a product is made of. This can be used to associate material properties with a product.

Attributes:
- *of_product*: reference to a **product_definition**
- *bulk_material*: reference to a **material**, specifying the substance or substances of which the bulk of the product is made

- *surface_material*: OPTIONAL reference to a **material**, specifying the substance or substances of which the outer surface of the product is made

Assertions:

- There shall be no more than one **product_material_relationship** per **product_definition**

4.2.15.3 material_property

A **material_property** is a specification of a valued property for a particular **material** in a particular **material_property_environment**.

Attributes:

- *of_material:* reference to a **material**
- *in_environment:* reference to a **material_property_environment**
- *property_value:* reference to a **function_valued_property** or a **string_valued_property** or a **number_valued_property**. It specifies the value of the property within *in_environment*.

Assertions:

- the **ato_case** indirectly referred to through *property* shal be the **space_mission_case** referred to by *in_environment*.

4.2.15.4 material_property_environment

A **material_property_environment** is a **space_mission_aspect** that captures a consistent set of environmental conditions which are relevant to material properties.

Attributes:

- None

Assertions:

- None

Notes:

NOTE 94 – The attributes and assertions are only those inherited from **space_mission_aspect**.

WORKING NOTE 12 – The concept is a slightly descoped version of **data_environment** as defined in [STEP-45].

Examples:

EXAMPLE 26 – “Beginning of Life”, “After 5000 Equivalent Sun Hours”, “End of Life”, “Uncontaminated” are all examples for the *name* of a **material_property_environment** in a thermal engineering context.

4.2.15.5 commercial_aspects_of_material

A **commercial_aspects_of_material** is a specification of commercial information about a **material**.

Attributes:

- *of_material:* reference to a **material**
- *commercial_name*: label string, specifying the commercial name under which the material is known
- *supplier*: reference to an organisation, which is the supplier of the material

4.2.16 additional_physical_quantities_and_measures UoF

All quantities and measures as defined in [STEP-NRF-AP] and [STEP-41] are applicable.

In addition the following shall be supported.

4.2.16.1 derived_unit

A **derived_unit** is an expression of units. With a **derived_unit** a particular combination of units can be specified.

EXAMPLE 27 – Watt per square meter (W/m^2) is a **derived_unit** that could be used to express the values of a thermal-radiative flux measure. In basic SI units this could also be expressed as kilogram per second cubed (kg/s^3) which is equivalent to W/m^2, but for human interpretation W/m^2 makes much more sense and should be conserved in an exchange dataset.

4.2.16.2 conversion_unit

A **conversion_unit** is a unit based on a multiplication factor with a measure expressed in another unit and possibly the addition of an offset expressed in the conversion unit itself.

Attributes:

- *name*: label string
- *factor*: **measure_with_unit**
- *offset*: **measure_with_unit**

Assertions:

- None

Examples:

EXAMPLE 28 – The **conversion_unit** for “inch” could have “Inch” as its *name*, 0.0254 meter as its *factor* and 0. meter as its *offset*.

EXAMPLE 29 – The **conversion_unit** for “Degree Fahrenheit” could have “degree_Fahrenheit” as its *name*, 9/5 degree_Celsius as its *factor* and 32 degree_Celsius as its *offset*.

4.2.16.3 function_valued_property

A **function_valued_property** is the definition of the value of a **property_class_scalar_functional** in the context of a **property_class_usage**. The value is specified as an index in the list of the **parameterized_functions** that are applicable for this **property_class_usage**.

Attributes:

- *valued_property*: reference the **property_class_usage** which is valued;
- *value*: index of a parameterized function chosen in the list of the functions applicable for *valued_property.*

Assertions:

- *valued_property.property* shall refer to a **property_class_scalar_functional**;
- there shall be exactly one **list_of_functions** such its attribute *applicable_for* contains*valued_property* in its set of values and such *value* is in the index range of the list.

4.2.16.4 string_valued_property

A **string_valued_property** is the definition of the value of a **property_class_scalar_descriptive** in the context of a **property_class_usage**. The value is specified as an index in the list of the strings that are applicable for this **property_class_usage**.

Attributes:

- *valued_property*: reference the **property_class_usage** which is valued;
- *value*: index of an element of the list of the strings applicable for *valued_property.*

Assertions:

- *valued_property.property* shall refer to a **property_class_scalar_descriptive**;
- there shall be exactly one **list_of_descriptive_values** such its attribute *applicable_for* contains*valued_property* in its set of values and such *value* is in the index range of the list.

4.2.16.5 number_valued_property

A **number_valued_property** is the specification of the value of a **property_class_scalar_quantitative** in the context of a **property_class_usage**.

Attributes:

- *valued_property*: reference the **property_class_usage** which is valued;

- *value*: NUMBER that specifies the value of the property.

Assertions:

- *valued_property.property* shall refer to a **property_class_scalar_quantitative**;

4.3 Application Assertions

NOTE 95 – Assertions which pertain directly to one application object are stated with its definition in clause 4.2.

Global assertions in addition to those specified in subclause 4.3 of [STEP-NRF-AP] are:

1) A **kinematic_model** can not be instantiated without a **geometric_model**.
2) A **basic_thermal_radiative_model** can not be instantiated without a **geometric_model**.
3) An **meshed_thermal_radiative_model** can not be instantiated without a **basic_thermal_radiative_model**.
4) A **primitive_bounded_surface** shall be referenced by a maximum of two **thermal_radiative_face_set**s per **basic_thermal_radiative_model**. If there are two **thermal_radiative_face_set**s, then their *active_side* attributes shall be one of the following combinations: (none, none), (none, positive), (none, negative) and (negative, positive).
5) For a given **material**, there shall be at least one **product_material_relationship** that refers to it as *bulk_material* or *surface_material*.

Clause 5 - Application Interpreted Model

5.1 Mapping Table

This clause contains the mapping table that shows how each application object defined in clause 4.2 of this Part maps to one or several AIM resource constructs (see subclause 5.2 “AIM EXPRESS short listing”).

The mapping is not provided in the tabular format required by [STEP-AP-Guide], but in the simpler serial format presented below. This was done to improve the efficiency of the mapping activity, while keeping exactly the same information. The mapping is formatted as follows:

For each application object:

ARM_OBJECT <application_object_name>
IS_MAPPED_TO
<aim_element> USED_FROM <resource_schema>
[WITH_RULES <rule_number>[,<rule_number>...]]
WITH_REFERENCE_PATH
<reference_path>

And then for each attribute of the application object:

ATTRIBUTE <application_object_attribute_name>
IS_MAPPED_TO
<aim_element> USED_FROM <resource_schema>
WITH_REFERENCE_PATH
<reference_path>

The different elements are explained as follows:

- The <application_object_name> is the name of an application element as it appears in the application object definition in subclause 4.2. Attribute names are listed after the application object to which they belong.
- The <aim_element> is the name of an AIM element as it appears in the AIM (subclause 5.2), or the term ‘IDENTICAL MAPPING’ or the term ‘PATH’. AIM entities are written in lower case. Attribute names are referred to as <entity_name>.<attribute_name>. The mapping of an application element may result in several related AIM elements. Each of these AIM elements will require a line of its own in the reference path section. The term ‘IDENTICAL MAPPING’ indicates that both application objects of an application assertion map to the same AIM element. The term ‘PATH’ indicates that the application assertion maps to the entire reference path.
- The USED_FROM <resource_schema> gives, for those AIM elements that are interpreted from the integrated resources, the number of the corresponding Part of ISO 10303. For those AIM elements that are incorporated from an application interpreted construct (AIC), this is the AIC identifier, as defined below.
  For those AIM elements that are created for the purpose of this AP, this is the acronym of the Space-domain Integrated Resources or this AP – i.e. “SIR” and “TAS” respectively.
- Where applicable, in the WITH_RULES <rule_number>[,<rule_number>...] contruct, one or several numbers may be given which refer to rules that apply to the current AIM element. For rules that are derived from relationships between application objects, the same rule is referred to by the mapping entries of all the involved AIM elements. The expanded names of the rules are listed after all mapping clauses.
- The WITH_REFERENCE_PATH <reference_path> construct is used to describe fully the mapping of an ARM element. It may be necessary to specify a reference path through several related AIM elements. A single AIM element is documented on a single line with a symbol which defines its relationship to the AIM element on the succeeding row in the column. The reference path column documents the role of an AIM element relative to the AIM element in the line succeeding it. Two or

more such related AIM elements define the interpretation of the Integrated Resources that satisfies the requirement specified by the application object. The reference path construct may be omitted when the path is trivial.

For each AIM element that has been created for use within this Part, a reference path up to its supertype from an integrated resource is specified.

- For the expression of reference paths and the relationships between AIM elements, the following notational conventions apply:

–>	attribute references an ENTITY or SELECT type given in the following row;
<–	attribute is an ENTITY or SELECT type referenced by the attribute in the following row;
=>	entity is a SUPERTYPE of the entity given in the following row;
<=	entity is a SUBTYPE of the entity given in the following row;
=	the type of the attribute is constrained to one of the types of the parent SELECT type, or, the value of the attribute is constrained to one specified string or to one specified choice in an ENUMERATION;
[]	all the enclosed sections in the reference path are required to satisfy an information requirement;
()	one of the enclosed sections in the reference path is required to map an information requirement;
<>	at least one of the enclosed sections of the reference path are required to map the information requirement;
{ }	enclosed section constrains the reference path to satisfy an information requirement;
| |	only the enclosed entity, and not its subtypes, may be instantiated;
*	any number of the relationship entity preceding the asterisk may be instantiated
[i]	attribute is an aggregation of which a single member is constrained;
[n]	attribute is an aggregation of which member n is constrained;

Notes:

NOTE 96 – The acronym NO MAPPING in the following tables means no resource has been found to map the application requirement,

NOTE 97 – Readers are reminded that the reference path gives only a summary of the entities to be instantiated to cover the requirement defined by an application object. It does not explicitate computations which may be needed to translate an attribute defined in the ARM into attributes of entities defined in the AIM.

5.1.1 Mapping Table - product_structure UoF

5.1.1.1 ARM_OBJECT product

IS_MAPPED_TO
product USED_FROM 41
WITH_RULES 18

5.1.1.2 ARM_OBJECT product_context

IS_MAPPED_TO
product_context USED_FROM 41
WITH_RULES 10

ATTRIBUTE name
IS_MAPPED_TO
application_context_element.name USED_FROM 41
WITH_REFERENCE_PATH
product_context<=application_context_element
application_context_element.name

ATTRIBUTE discipline_type
IS_MAPPED_TO
product_context.discipline_type USED_FROM 41

5.1.1.3 ARM_OBJECT product_definition

IS_MAPPED_TO
product_definition USED_FROM 41
WITH_RULES 17

5.1.1.4 ARM_OBJECT product_definition_context

IS_MAPPED_TO
product_definition_context USED_FROM 41
WITH_RULES 9

5.1.1.5 ARM_OBJECT product_next_assembly_usage_relationship

IS_MAPPED_TO
next_assembly_usage_occurence USED_FROM 44
WITH_RULES 12

ATTRIBUTE id
IS_MAPPED_TO
product_definition_relationship.id USED_FROM 41
WITH_REFERENCE_PATH
next_assembly_usage_occurence <= assembly_component_usage
assembly_component_usage <= product_definition_usage
product_definition_usage <= product_definition_relationship
product_definition_relationship.id

product_next_assembly_usage_relationship to product_definition (as assembly)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
next_assembly_usage_occurence <= assembly_component_usage
assembly_component_usage <= product_definition_usage
product_definition_usage <= product_definition_relationship
product_definition_relationship.relating_product_definition -> product_definition

product_next_assembly_usage_relationship to product_definition (as constituent)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
next_assembly_usage_occurence <= assembly_component_usage
assembly_component_usage <= product_definition_usage
product_definition_usage <= product_definition_relationship
product_definition_relationship.related_product_definition -> product_definition

5.1.1.6 ARM_OBJECT product_version

IS_MAPPED_TO
SIR_product_version USED_FROM SIR
WITH_RULES 13

ATTRIBUTE id
IS_MAPPED_TO
product_definition_formation.id USED_FROM 41
WITH_REFERENCE_PATH
SIR_product_version <= product_definition_formation
product_definition_formation.id

ATTRIBUTE description
IS_MAPPED_TO
product_definition_formation.description USED_FROM 41
WITH_REFERENCE_PATH

SIR_product_version <= product_definition_formation
product_definition_formation.description

product_version to product (as of_product)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_product_version <= product_definition_formation
product_definition_formation.of_product -> product

product_version to product_definition (as definitions)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_product_version <= product_definition_formation
product_definition_formation <- product_definition.formation

5.1.2 Mapping Table - network_model_representation UoF

5.1.2.1 ARM_OBJECT model_represents_product_relationship

IS_MAPPED_TO
TAS_model_product_relationship USED_FROM TAS
WITH_RULES 17

model_represents_product_relationship to network_model (as model_constituent)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_model_product_relationship <= property_definition_representation
property_definition_representation.used_representation -> representation
representation => SIR_model

model_represents_product_relationship to network_model_constituent (as model_constituent)
#1 if the constituent is a network_node
#2 if the constituent is a network_node_relationship
#3 if the constituent is a submodel_usage
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_model_product_relationship <= property_definition_representation
property_definition_representation.used_representation -> representation
representation.items[1] -> network_model_constituent
#1 (network_model_constituent = SIR_network_node)
#2 (network_model_constituent = SIR_network_node_relationship)
#3 (network_model_constituent = SIR_submodel_usage)

model_represents_product_relationship to product_definition (as represented_product)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
property_definition_representation.definition -> property_definition
{property_definition.description = 'shape description for space mission case'}
property_definition.definition = characterized_definition
(characterized_definition = characterized_product_definition
characterized_product_definition = product_definition)
(characterized_definition = shape_definition
(shape_definition = product_definition_shape)
(shape_definition = shape_aspect))

5.1.2.2 ARM_OBJECT network_model

IS_MAPPED_TO
SIR_model USED_FROM SIR
WITH_RULES 11, 3, 17

ATTRIBUTE id
IS_MAPPED_TO
SIR_model.id USED_FROM SIR

ATTRIBUTE version_id
IS_MAPPED_TO
SIR_model.version_id USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
representation.name USED_FROM 43
WITH_REFERENCE_PATH
SIR_model <= representation
representation.name

ATTRIBUTE description
IS_MAPPED_TO
SIR_model.description USED_FROM SIR

network_model to security_classification_level (as security_class)
IS_MAPPED_TO
PATH
WITH_RULES 36
WITH_REFERENCE_PATH
SIR_model = security_classified_item
security_classified_item <- NRF_security_assignment.items[i]
NRF_security_assignment <= security_classification_assignment
security_classification_assignment.assigned_security_classification -> security_classification
security_classification.security_level -> security_classification_level

network_model to network_model_constituent (as constituents)
#1 if the constituent is a network_node
#2 if the constituent is a network_node_relationship
#3 if the constituent is a submodel_usage
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_model <= representation
representation.items[i] -> network_model_constituent
#1 (network_model_constituent = SIR_network_node)
#2 (network_model_constituent = SIR_network_node_relationship)
#3 (network_model_constituent = SIR_submodel_usage)

5.1.2.3 ARM_OBJECT network_node

IS_MAPPED_TO
SIR_node USED_FROM SIR

ATTRIBUTE id
IS_MAPPED_TO
SIR_node.id USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
SIR_node <= representation_item
representation_item.name

network_node to network_node_class (as class)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_node.classes[i] -> type_qualifier

network_node to security_classification_level (as security_class)
IS_MAPPED_TO

PATH
WITH_RULES 36
WITH_REFERENCE_PATH
SIR_node = security_classified_item
security_classified_item <- NRF_security_assignment.items[i]
NRF_security_assignment <= security_classification_assignment
security_classification_assignment.assigned_security_classification -> security_classification
security_classification.security_level -> security_classification_level

5.1.2.4 ARM_OBJECT network_node_class

IS_MAPPED_TO
type_qualifier USED_FROM 45
WITH_RULES 21

ATTRIBUTE name
IS_MAPPED_TO
type_qualifier.name USED_FROM 45

5.1.2.5 ARM_OBJECT network_node_relationship

IS_MAPPED_TO
SIR_node_relationship USED_FROM SIR

ATTRIBUTE id
IS_MAPPED_TO
SIR_node_relationship.id USED_FROM 43

ATTRIBUTE name
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
SIR_node_relationship <= representation_item
representation_item.name

network_node_relationship to network_node_relationship_class (as class)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR _node_relationship.classes[i] -> type_qualifier

network_node_relationship to security_classification_level (as security_class)
IS_MAPPED_TO
PATH
WITH_RULES 36
WITH_REFERENCE_PATH
SIR_node_relationship = security_classified_item
security_classified_item <- NRF_security_assignment.items[i]
NRF_security_assignment <= security_classification_assignment
security_classification_assignment.assigned_security_classification -> security_classification
security_classification.security_level -> security_classification_level

network_node_relationship to network_node (as nodes)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_node_relationship.nodes[i] -> SIR_node_or_usage
SIR_node_or_usage = SIR_node

network_node_relationship to network_node_usage (as nodes)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_node_relationship.nodes[i] -> SIR_node_or_usage
SIR_node_or_usage = SIR_node_usage

5.1.2.6 ARM_OBJECT network_node_relationship_class

IS_MAPPED_TO
type_qualifier USED_FROM 45
WITH_RULES 21

ATTRIBUTE name
IS_MAPPED_TO
type_qualifier.name USED_FROM 45

5.1.2.7 ARM_OBJECT network_node_relationship_usage

IS_MAPPED_TO
SIR_node_relationship_usage USED_FROM SIR

network_node_relationship_usage to submodel_usage (as used_submodel)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_node_relationship_usage.used_submodel -> SIR_submodel_usage

network_node_relationship_usage to network_node_relationship (as relationship)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_node_relationship_usage.component -> SIR_node_relationship

5.1.2.8 ARM_OBJECT network_node_usage

IS_MAPPED_TO
SIR_node_usage USED_FROM SIR

network_node_usage to submodel_usage (as used_submodel)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_node_usage.used_submodel -> SIR_submodel_usage

network_node_usage to network_node (as node)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_node_usage.node -> SIR_node

5.1.2.9 ARM_OBJECT submodel_usage

IS_MAPPED_TO
SIR_submodel_usage USED_FROM SIR

ATTRIBUTE id
IS_MAPPED_TO
SIR_submodel_usage.id USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
SIR_submodel_usage <= mapped_item
mapped_item <= representation_item
representation_item.name

submodel_usage to network_model (as submodel)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_submodel_usage <= mapped_item
mapped_item.mapping_source -> representation_map
representation_map.mapped_representation -> representation

representation => SIR_model

submodel_usage to network_node (as source_node)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_submodel_usage <= mapped_item
mapped_item.mapping_source -> representation_map
representation_map.mapping_origin -> representation_item
representation_item => SIR_node

submodel_usage to network_node (as target_node)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_submodel_usage <= mapped_item
mapped_item.mapping_target -> representation_item
representation_item => SIR_node

5.1.3 Mapping Table - bulk_results UoF

5.1.3.1 ARM_OBJECT aspect

IS_MAPPED_TO
SIR_aspect USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
SIR_aspect.name USED_FROM SIR

aspect to property_class_usage (as for_property_class)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_aspect.valued_property -> SIR_property_usage

ATTRIBUTE lifetime
IS_MAPPED_TO
SIR_aspect.lifetime USED_FROM SIR

aspect to model_component_sequence (as component_sequence)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_aspect.component_sequence -> SIR_component_sequence

aspect to scan (as scans)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_aspect.scans[i] -> SIR_scan

aspect to security_classification_level (as security_class)
IS_MAPPED_TO
PATH
WITH_RULES 4
WITH_REFERENCE_PATH
SIR_aspect = security_classified_item
security_classified_item <- NRF_security_assignment.items[i]
NRF_security_assignment <= security_classification_assignment
security_classification_assignment.assigned_security_classification -> security_classification
security_classification.security_level -> security_classification_level

5.1.3.2 ARM_OBJECT ato_campaign

IS_MAPPED_TO
SIR_ato_campaign USED_FROM SIR

ato_campaign to organizational_project (as of_project)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_ato_campaign.of_project -> organizational_project

ATTRIBUTE name
IS_MAPPED_TO
group.name USED_FROM 41
WITH_REFERENCE_PATH
SIR_ato_campaign <= group
group.name

ATTRIBUTE description
IS_MAPPED_TO
group.description USED_FROM 41
WITH_REFERENCE_PATH
SIR_ato_campaign <= group
group.description

ato_campaign to date_and_time (as creation_date_and_time)
IS_MAPPED_TO
PATH
WITH_RULES 39
WITH_REFERENCE_PATH
SIR_ato_campaign = dated_item
dated_item<- NRF_date_assignment.dated_items[i]
NRF_date_assignment <= date_and_time_assignment
date_and_time_assignment.assigned_date -> date_and_time
{date_and_time_assignment.role -> date_time_role
date_time_role.name = ’CREATION_DATE’}

ato_campaign to date_and_time (as last_modification_date_and_time)
IS_MAPPED_TO
PATH
WITH_RULES 37
WITH_REFERENCE_PATH
SIR_ato_campaign = dated_item
dated_item<- NRF_date_assignment.dated_items[i]
NRF_date_assignment <= date_and_time_assignment
date_and_time_assignment.assigned_date -> date_and_time
{date_and_time_assignment.role -> date_time_role
date_time_role.name = ’LAST_MODIFICATION_DATE’}

ato_campaign to person_and_organization (as contact_person_organisation)
IS_MAPPED_TO
PATH
WITH_RULES 38
WITH_REFERENCE_PATH
SIR_ato_campaign = sourced_item
sourced_item <- NRF_contact_assignment.items[i]
NRF_contact_assignment <= person_and_organization_assignment
person_and_organization_assignment.assigned_person_and_organization -> person_and_organization
person_and_organization_assignment.role -> person_and_organization_role
{person_and_organization_role.name = ’CONTACT_PERSON/ORGANIZATION’}

ato_campaign to approval (as approvals)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_ato_campaign = approved_item
approved_item <- NRF_approval_assignment.approved_items[i]
NRF_approval_assignment.assigned_approval -> approval
approval.status -> approval_status

ato_campaign to security_classification_level (as security_class)
IS_MAPPED_TO
PATH
WITH_RULES 36
WITH_REFERENCE_PATH
SIR_ato_campaign = security_classified_item
security_classified_item <- NRF_security_assignment.items[i]
NRF_security_assignment <= security_classification_assignment
security_classification_assignment.assigned_security_classification -> security_classification
security_classification.security_level -> security_classification_level

ATTRIBUTE application_interpreted_model_schema_name
IS_MAPPED_TO
application_protocol_definition.application_interpreted_model_schema_name USED_FROM 41
WITH_RULES 35
WITH_REFERENCE_PATH
SIR_ato_campaign.in_context -> application_context_element
application_context_element.frame_of_reference -> application_context
application_context <- application_protocol.application

ATTRIBUTE application_protocol_year
IS_MAPPED_TO
application_protocol_definition.application_protocol_year USED_FROM 41
WITH_REFERENCE_PATH
SIR_ato_campaign.in_context -> application_context_element
application_context_element.frame_of_reference -> application_context
application_context <- application_protocol.application
application_protocol_definition.application_protocol_year

5.1.3.3 ARM_OBJECT ato_case

IS_MAPPED_TO
SIR_ato_case USED_FROM SIR

ato_case to ato_campaign (as of_campaign)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_ato_case.campaign_membership -> SIR_case_in_campaign
SIR_case_in_campaign <= group_assignment
group_assignment.assigned_group -> group
group => SIR_ato_campaign

ATTRIBUTE id
IS_MAPPED_TO
SIR_ato_case.id USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
SIR_ato_case.name USED_FROM SIR

ATTRIBUTE description
IS_MAPPED_TO
SIR_ato_case.description USED_FROM SIR

ato_case to network_model (as root)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_ato_case.root_model -> SIR_model

ato_case to property_class_usage (as abscissa_class)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_ato_case.abscissa_class -> SIR_property_usage

ATTRIBUTE abscissa_sequencing
IS_MAPPED_TO
SIR_ato_case.abscissa_sequencing USED_FROM SIR

ato_case to aspect (as predefined_aspects)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_ato_case = SIR_case_or_phase
SIR_case_or_phase <- SIR_initial_conditioning.conditioned_data
SIR_initial_conditioning.initial_condition -> SIR_aspect

ato_case to security_classification_level (as security_class)
IS_MAPPED_TO
PATH
WITH_RULES 36
WITH_REFERENCE_PATH
SIR_ato_case = security_classified_item
security_classified_item <- NRF_security_assignment.items[i]
NRF_security_assignment <= security_classification_assignment
security_classification_assignment.assigned_security_classification -> security_classification
security_classification.security_level -> security_classification_level

5.1.3.4 ARM_OBJECT ato_phase

IS_MAPPED_TO
SIR_ato_phase USED_FROM SIR

ATTRIBUTE id
IS_MAPPED_TO
SIR_ato_phase.id USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
SIR_ato_phase.name USED_FROM SIR

ATTRIBUTE description
IS_MAPPED_TO
SIR_ato_phase.description USED_FROM SIR

ato_phase to ato_case (as of_case)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_ato_phase.of_case -> SIR_ato_case

ato_phase to aspect (as predefined_aspects)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_ato_phase = SIR_case_or_phase
SIR_case_or_phase <- SIR_initial_conditioning.conditioned_data
SIR_initial_conditioning.initial_condition -> SIR_aspect

ato_phase to security_classification_level (as security_class)
IS_MAPPED_TO
PATH
WITH_RULES 4
WITH_REFERENCE_PATH
SIR_ato_phase = security_classified_item
security_classified_item <- NRF_security_assignment.items[i]
NRF_security_assignment <= security_classification_assignment
security_classification_assignment.assigned_security_classification -> security_classification
security_classification.security_level -> security_classification_level

5.1.3.5 ARM_OBJECT list_of_descriptive_values

IS_MAPPED_TO
SIR_list_of_descriptive_values USED_FROM SIR

list_of_descriptive_values to property_class_usage (as applicable_for)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_list_of_ descriptive_values.applicable_for[i] -> SIR_applicability_of_values
SIR_applicability_of_values.valid_usage -> SIR_property_usage

ATTRIBUTE name
IS_MAPPED_TO
SIR_list_of_descriptive_values.name USED_FROM SIR

ATTRIBUTE entries
IS_MAPPED_TO
SIR_list_of_descriptive_values.entries USED_FROM SIR

5.1.3.6 ARM_OBJECT list_of_functions

IS_MAPPED_TO
SIR_list_of_functions USED_FROM SIR

list_of_functions to property_class_usage (as applicable_for)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_list_of_ functions.applicable_for[i] -> SIR_applicability_of_values
SIR_applicability_of_values.valid_usage -> SIR_property_usage

ATTRIBUTE name
IS_MAPPED_TO
SIR_list_of_functions.name USED_FROM SIR

list_of_functions to parameterized_functions (as entries)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_list_of_functions.entries[i] -> SIR_parameterized_function

5.1.3.7 ARM_OBJECT model_component_sequence

IS_MAPPED_TO
SIR_component_sequence USED_FROM SIR

ATTRIBUTE id
IS_MAPPED_TO
SIR_component_sequence.id USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
SIR_component_sequence.name USED_FROM SIR
WITH_RULES 29

model_component_sequence to network_model_component (as components)
IS_MAPPED_TO
#1 if the constituent is a network_node
#2 if the constituent is a network_node_relationship
#3 if the constituent is a submodel_usage
#4 if the constituent is a network_node_usage
#5 if the constituent is a network_node_relationship_usage
#6 if the constituent is a network_model
PATH
WITH_REFERENCE_PATH
NRF_component_sequence.components[i] -> SIR_network_component
#1 (SIR_network_component = SIR_node)

#2 (SIR_network_component = SIR_node_relationship)
#3 (SIR_network_component = SIR_submodel_usage)
#4 (SIR_network_component = SIR_node_usage)
#5 (SIR_network_component = SIR_node_relationship_usage)
#6 (SIR_network_component = SIR_model)

5.1.3.8 ARM_OBJECT property_class

IS_MAPPED_TO
SIR_property_class USED_FROM SIR

5.1.3.9 ARM_OBJECT property_class_scalar

IS_MAPPED_TO
SIR_property_scalar USED_FROM SIR

property_class_scalar to property_name (as name)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_property_scalar <= SIR_property_class
SIR_property_class.name -> SIR_property_name

ATTRIBUTE symmetry
IS_MAPPED_TO
SIR_property_scalar.symmetry USED_FROM SIR

5.1.3.10 ARM_OBJECT property_class_scalar_descriptive

IS_MAPPED_TO
SIR_property_descriptive USED_FROM SIR

5.1.3.11 ARM_OBJECT property_class_scalar_functional

IS_MAPPED_TO
SIR_property_functional USED_FROM SIR

5.1.3.12 ARM_OBJECT property_class_scalar_quantitative

IS_MAPPED_TO
SIR_property_quantitative USED_FROM SIR

5.1.3.13 ARM_OBJECT property_class_tensor

IS_MAPPED_TO
SIR_property_tensor USED_FROM SIR

property_class_tensor to property_name (as name)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_property_tensor <= SIR_property_class
SIR_property_class.name -> SIR_property_name

ATTRIBUTE tensor_order
IS_MAPPED_TO
SIR_property_tensor.order USED_FROM SIR

ATTRIBUTE dimensionality
IS_MAPPED_TO
SIR_property_tensor.dimensionality USED_FROM SIR

5.1.3.14 ARM_OBJECT property_class_usage

IS_MAPPED_TO
SIR_property_usage USED_FROM SIR

property_class_usage to ato_case (as case)
IS_MAPPED_TO

PATH
WITH_REFERENCE_PATH
SIR_property_usage.case -> SIR_ato_case

property_class_usage to property_class (as property)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_property_usage.property -> SIR_property_class

property_class_usage to property_role (as meaning)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_property_usage.meaning -> SIR_property_meaning
SIR_property_meaning.roles[i] -> type_qualifier

5.1.3.15 ARM_OBJECT property_name

IS_MAPPED_TO
SIR_property_name USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
SIR_property_name.name USED_FROM SIR

5.1.3.16 ARM_OBJECT property_role

IS_MAPPED_TO
type_qualifier USED_FROM 45
WITH_RULES 21

ATTRIBUTE name
IS_MAPPED_TO
type_qualifier.name USED_FROM 45

5.1.3.17 ARM_OBJECT run

IS_MAPPED_TO
NRF_run USED_FROM TAS
WITH_REFERENCE_PATH
NRF_run <= executed_action

run to ato_phase (as of_phase)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
NRF_run <= executed_action
executed_action <= action
action <- action_assignment.assigned_action
action_assignment => NRF_run_phase
NRF_run_phase.subject -> SIR_ato_phase

ATTRIBUTE id
IS_MAPPED_TO
action.name USED_FROM 41
WITH_REFERENCE_PATH
executed_action <= action
action.name

ATTRIBUTE name
IS_MAPPED_TO
name_assignment.assigned_name USED_FROM 41
WITH_REFERENCE_PATH
NRF_run = named_item
named_item <- TAS_name_assignment.named_items[i]
TAS_name_assignment <= name_assignment

name_assignment.assigned_name

ATTRIBUTE description
IS_MAPPED_TO
action.description USED_FROM 41
WITH_REFERENCE_PATH
executed_action <= action
action.description

run to date_and_time (as timestamp)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
NRF_run = dated_item
dated_item<- TAS_date_assignment.dated_items[i]
TAS_date_assignment <= date_and_time_assignment
date_and_time_assignment.assigned_date_and_time -> date_and_time
date_and_time_assignment.role -> date_time_role
{date_time_role.name = 'start date and time'}

ATTRIBUTE creator_tool_or_facility
IS_MAPPED_TO
action_resource.name USED_FROM 41
WITH_REFERENCE_PATH
executed_action <= action
action = supported_item
supported_item <- action_resource.usage[1]
{action_resource.kind -> action_resource_type
action_resource_type = 'tool or facility'}
action_resource.name

run to aspect (as aspects)
IS_MAPPED_TO
PATH
WITH_RULES 20
WITH_REFERENCE_PATH
NRF_run <= executed_action
executed_action <= action
action <- action_assignment.assigned_action
action_assignment => NRF_result_assignment
NRF_run_results.results[i] -> SIR_aspect

5.1.3.18 ARM_OBJECT run_sequence

IS_MAPPED_TO
NRF_run_sequence USED_FROM TAS
WITH_RULES 24
WITH_REFERENCE_PATH
NRF_run_sequence <= action

run_sequence to ato_case (as of_case)
IS_MAPPED_TO
PATH
WITH_RULES 20
WITH_REFERENCE_PATH
NRF_run_sequence <= action
action <- action_assignment.assigned_action
action_assignment => NRF_run_sequence_case
NRF_run_sequence_case.subject -> SIR_ato_case

ATTRIBUTE id
IS_MAPPED_TO
action.name USED_FROM 41

ATTRIBUTE name
IS_MAPPED_TO

name_assignment.assigned_name USED_FROM 41
WITH_REFERENCE_PATH
NRF_run_sequence = named_item
named_item <- TAS_name_assignment.named_items[i]
TAS_name_assignment <= name_assignment
name_assignment.assigned_name

ATTRIBUTE description
IS_MAPPED_TO
action.description USED_FROM 41
WITH_REFERENCE_PATH
NRF_run_sequence <= action
action.description

run_sequence to run (as runs)
IS_MAPPED_TO
PATH
WITH_RULES 25
WITH_REFERENCE_PATH
NRF_run_sequence <= action
action <- action_relationship.relating_action
action_relationship.related_action -> action
action => executed_action
executed_action => NRF_run

5.1.3.19 ARM_OBJECT scalar_in_tensor

IS_MAPPED_TO
SIR_scalar_in_tensor USED_FROM SIR

scalar_in_tensor to property_class_tensor (as tensor)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_scalar_in_tensor.tensor -> SIR_property_tensor

scalar_in_tensor to property_class_scalar (as scalar)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_scalar_in_tensor.scalar -> SIR_property_scalar

ATTRIBUTE position
IS_MAPPED_TO
SIR_scalar_in_tensor.position USED_FROM SIR

ATTRIBUTE roles
IS_MAPPED_TO
type_qualifier USED_FROM 45
WITH_REFERENCE_PATH
SIR_scalar_in_tensor.roles -> SIR_property_meaning
SIR_property_meaning.roles[i] -> type_qualifier

5.1.3.20 ARM_OBJECT scan

IS_MAPPED_TO
SIR_scan USED_FROM SIR
WITH_RULES 21

scan to aspect (as of_aspect)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_scan <- SIR_aspect.scans[i]

ATTRIBUTE abscissa_value
IS_MAPPED_TO

SIR_scan.abscissa_value USED_FROM SIR

scan to scan_mask (as mask)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_scan.mask -> SIR_scan_mask

ATTRIBUTE values
IS_MAPPED_TO
SIR_scan.values USED_FROM SIR

5.1.3.21 ARM_OBJECT scan_derivation_procedure

IS_MAPPED_TO
NRF_scan_derivation_procedure USED_FROM NRF
WITH_REFERENCE_PATH
NRF_scan_derivation_procedure <= action_method

ATTRIBUTE name
IS_MAPPED_TO
action_method.name USED_FROM 41

5.1.3.22 ARM_OBJECT scan_mask

IS_MAPPED_TO
SIR_mask USED_FROM SIR

ATTRIBUTE defined_ranges
IS_MAPPED_TO
SIR_mask.defined_ranges USED_FROM SIR

5.1.3.23 ARM_OBJECT scan_of_derived_values

IS_MAPPED_TO
NRF_scan_derived USED_FROM SIR
WITH_REFERENCE_PATH
NRF_scan_derived <= SIR_scan

scan_of_derived_values to scan_derivation_procedure (as derivation_procedure)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
NRF_scan_derived <- NRF_derivation_result.result
NRF_derivation <= action_assignment
action_assignment.assigned_action -> action
action.chosen_method -> action_method
action_method => NRF_derivation_procedure

ATTRIBUTE abscissa_start
IS_MAPPED_TO
NRF_derivation_bounds.bounds[1] USED_FROM SIR
WITH_REFERENCE_PATH
NRF_scan_derived <- NRF_derivation_result.result
NRF_derivation <= action_assignment
action_assignment.assigned_action -> action
action => NRF_derivation
NRF_derivation <- action_assignment.assigned_action
action_assignment => NRF_derivation_bounds
NRF_derivation_bounds.bounds[1]

ATTRIBUTE abscissa_end
IS_MAPPED_TO
NRF_derivation_bounds.bounds[2] USED_FROM SIR
WITH_REFERENCE_PATH
NRF_scan_derived <- NRF_derivation_result.result
NRF_derivation <= action_assignment

action_assignment.assigned_action -> action
action => NRF_derivation
NRF_derivation <- action_assignment.assigned_action
action_assignment => NRF_derivation_bounds
NRF_derivation_bounds.bounds[2]

5.1.3.24 ARM_OBJECT scan_of_sampled_values

IS_MAPPED_TO
NRF_scan_sampled USED_FROM SIR
WITH_REFERENCE_PATH
NRF_scan_sampled <= SIR_scan

5.1.3.25 ARM_OBJECT unit_assignment

IS_MAPPED_TO
SIR_unit_assignment USED_FROM SIR

unit_assignment to ato_case (as of_case)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_unit_assignment.in_case -> SIR_ato_case

unit_assignment to property_class_scalar (as for_property_class)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_unit_assignment.for_property -> SIR_property_scalar

unit_assignment to unit (as assigned_unit)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_unit_assignment.assigned_unit -> unit

5.1.4 Mapping Table - parameterized_function UoF

5.1.4.1 ARM_OBJECT cyclic_tabular_function

IS_MAPPED_TO
SIR_cyclic_tabular_function USED_FROM SIR

ATTRIBUTE period
IS_MAPPED_TO
SIR_cyclic_tabular_function.period USED_FROM SIR

5.1.4.2 ARM_OBJECT parameterized_function

IS_MAPPED_TO
SIR_parameterized_function USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
SIR_parameterized_function.name USED_FROM SIR

ATTRIBUTE description
IS_MAPPED_TO
SIR_parameterized_function.description USED_FROM SIR

parameterized_function to property_class_scalar (as parameter_properties)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_parameterized_function.parameters[i] -> SIR_variable

parameterized_function to property_class_scalar (as result_property)
IS_MAPPED_TO

PATH
WITH_REFERENCE_PATH
SIR_parameterized_function.result -> SIR_variable

parameterized_function to document_usage_constraint (as source)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_parameterized_function = documented_item
documented_item <- TAS_document_reference.documented_items[i]
TAS_document_reference <= document_reference
document_reference.assigned_document -> document
document <- document_usage_constraint.source

parameterized_function to date_and_time (as definition_timestamp)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_parameterized_function = dated_item
dated_item <- TAS_date_assignment.dated_items[i]
TAS_date_assignment <= date_and_time_assignment
date_and_time_assignment.assigned_date -> date_and_time
{date_and_time_assignment.role -> date_time_role
date_time_role.name = 'CREATION_DATE'}

5.1.4.3 ARM_OBJECT parameterized_function_with_formula

IS_MAPPED_TO
SIR_parameterized_function USED_FROM SIR

ATTRIBUTE formula
IS_MAPPED_TO
mathematical_string.text_representation USED_FROM 13584-42
WITH_REFERENCE_PATH
SIR_parameterized_function.formula -> mathematical_string
mathematical_string.text_representation

ATTRIBUTE language
IS_MAPPED_TO
NO MAPPING

ATTRIBUTE creator_tool_or_facility
IS_MAPPED_TO
NO MAPPING

5.1.4.4 ARM_OBJECT polynomial_function

IS_MAPPED_TO
SIR_polynomial_function USED_FROM SIR

ATTRIBUTE degree
IS_MAPPED_TO
int_literal USED_FROM 13584-20

ATTRIBUTE coefficients
IS_MAPPED_TO
real_litteral USED_FROM 13584-20

ATTRIBUTE lower_domain_bounds
IS_MAPPED_TO
[comparison_less_equal] USED_FROM 13584-20
[literal_number] USED_FROM 13584-20
WITH_REFERENCE_PATH
SIR_polynomial_function.lower_bound_expressions[i] -> less_or_greater
less_or_greater = comparison_less_equal
comparison_less_equal <= comparison_expression
comparison_expression <= binary_gen_expression

binary_gen_expression.operands[1] -> generic_expression
generic_expression => generic_literal
generic_literal => literal_number

ATTRIBUTE upper_domain_bounds
IS_MAPPED_TO
[comparison_greater_equal] USED_FROM 13584-20
[literal_number] USED_FROM 13584-20
WITH_REFERENCE_PATH
SIR_polynomial_function. upper_bounds_expressions[i] -> less_or_greater
less_or_greater = comparison_greater_equal
comparison_greater_equal <= comparison_expression
comparison_expression <= binary_gen_expression
binary_gen_expression.operands[1] -> generic_expression
generic_expression => generic_literal
generic_literal => literal_number

5.1.4.5 ARM_OBJECT tabular_function

IS_MAPPED_TO
SIR_tabular_function USED_FROM SIR

ATTRIBUTE parameter_values
IS_MAPPED_TO
list_ascending_real_literals USED_FROM SIR
WITH_REFERENCE_PATH
SIR_tabular_function.parameter_values[i] -> list_ascending_real_literals

ATTRIBUTE function_values
IS_MAPPED_TO
list_of_real_literals USED_FROM SIR
WITH_REFERENCE_PATH
SIR_tabular_function.result_values -> SIR_list_of_real_literal

ATTRIBUTE interpolation_type
IS_MAPPED_TO
SIR_tabular_function.interpolation_type USED_FROM SIR

ATTRIBUTE interpolation_degree
IS_MAPPED_TO
SIR_tabular_function.interpolation_degree USED_FROM SIR

5.1.5 Mapping Table - date_and_time UoF

5.1.5.1 ARM_OBJECT calendar_date

IS_MAPPED_TO
calendar_date USED_FROM 41

5.1.5.2 ARM_OBJECT coordinated_universal_time_offset

IS_MAPPED_TO
coordinated_universal_time_offset USED_FROM 41

5.1.5.3 ARM_OBJECT date

IS_MAPPED_TO
date USED_FROM 41
WITH_RULES 14, 26

5.1.5.4 ARM_OBJECT date_and_time

IS_MAPPED_TO
date_and_time USED_FROM 41

5.1.5.5 ARM_OBJECT local_time

IS_MAPPED_TO

local_time USED_FROM 41

5.1.6 Mapping Table - general_support UoF

5.1.6.1 ARM_OBJECT address

IS_MAPPED_TO
address USED_FROM 41

5.1.6.2 ARM_OBJECT approval

IS_MAPPED_TO
approval_status USED_FROM 41
WITH_RULES 28, 30

ATTRIBUTE level
IS_MAPPED_TO
approval_status.name USED_FROM 41
WITH_RULES 27

approval to date_and_time (as date_time)
IS_MAPPED_TO
PATH
WITH_RULES 33, 34
WITH_REFERENCE_PATH
aproval_status <- approval.status
approval <- approval_date_time.dated_approval
approval_date_time.date_time -> date_time_select
date_time_select = date_and_time

approval to person_and_organization (as by_person_organization)
IS_MAPPED_TO
PATH
WITH_RULES 32
WITH_REFERENCE_PATH
aproval_status <- approval.status
approval <- approval_person_organization.authorized_approval
approval_person_organization.person_organization -> person_organization_select
person_organization_select = person_and_organization

5.1.6.3 ARM_OBJECT document

IS_MAPPED_TO
document USED_FROM 41

5.1.6.4 ARM_OBJECT document_type

IS_MAPPED_TO
document_type USED_FROM 41

5.1.6.5 ARM_OBJECT document_usage_constraint

IS_MAPPED_TO
document_usage_constraint USED_FROM 41

5.1.6.6 ARM_OBJECT organization

IS_MAPPED_TO
organization USED_FROM 41

5.1.6.7 ARM_OBJECT organization_address

IS_MAPPED_TO
organizational_address USED_FROM 41

5.1.6.8 ARM_OBJECT organizational_project

IS_MAPPED_TO

organizational_project USED_FROM 41

5.1.6.9 ARM_OBJECT person

IS_MAPPED_TO
person USED_FROM 41
WITH_RULES 24

5.1.6.10 ARM_OBJECT person_and_organization

IS_MAPPED_TO
person_and_organization USED_FROM 41

5.1.6.11 ARM_OBJECT personal_address

IS_MAPPED_TO
personal_address USED_FROM 41

5.1.6.12 ARM_OBJECT security_classification_level

IS_MAPPED_TO
WITH_RULES 36, 22
security_classification_level USED_FROM 41

5.1.7 Mapping Table - space_mission_aspects UoF

5.1.7.1 ARM_OBJECT space_mission

IS_MAPPED_TO
SIR_ato_campaign USED_FROM SIR

5.1.7.2 ARM_OBJECT space_mission_case

IS_MAPPED_TO
SIR_space_mission_case USED_FROM SIR

space_mission_case to space_coordinate_system (as coordinate_system)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_space_mission_case.coordinate_system -> geometric_representation_context
geometric_representation_context => TAS_space_coordinate_system

5.1.7.3 ARM_OBJECT space_mission_phase

IS_MAPPED_TO
SIR_space_mission_phase USED_FROM SIR

space_mission_phase to space_mission_event (as start_event)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_space_mission_phase.bounding_events -> SIR_event_effectivity
SIR_event_effectivity.start_event

space_mission_phase to space_mission_event (as end_event)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_space_mission_phase.bounding_events -> SIR_event_effectivity
SIR_event_effectivity.start_event

space_mission_phase to space_mission_aspect (as space_aspects)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_space_mission_phase.aspects -> SIR_space_aspect_in_phase
SIR_space_aspect_in_phase.occurrence -> SIR_space_mission_aspect

5.1.7.4 ARM_OBJECT space_mission_aspect

IS_MAPPED_TO
SIR_space_mission_aspect USED_FROM SIR
WITH_RULES 7

space_mission_aspect to space_mission_case (as of_case)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_space_mission_aspect.of_case -> SIR_space_mission_case

ATTRIBUTE id
IS_MAPPED_TO
SIR_space_mission_aspect.id USED_FROM SIR

ATTRIBUTE name
#1 (if it is thermal_environment or a material_property_environment)
#2 (if it is a main_body_placement)
#3 (if it is an orbit_arc)
#4 (if it is a kinematic_articulation)
IS_MAPPED_TO
#1 (data_environment.name) USED_FROM 45
#2 (functionally_defined_transformation.name) USED_FROM 43
#3 (characterized_object.name) USED_FROM 41
#4 (pair_actuator.name) USED_FROM 105
WITH_REFERENCE_PATH
#1 (SIR_space_mission_aspect.characterized_object -> considered_aspect
considered_aspect = data_envrionment
data_environment.name)
#2 (SIR_space_mission_aspect.characterized_object -> considered_aspect
considered_aspect = SIR_placement_transformation
SIR_placement_transformation <= functionally_defined_transformation
functionally_defined_transformation.name)
#3 (SIR_space_mission_aspect.characterized_object -> considered_aspect
considered_aspect = property_definition_representation
{property_definition_representation => TAS_orbit_arc}
property_definition_representation.definition -> property_definition
property_definition.definition = characterized_definition
characterized_definition = characterized_object
characterized_object.name)
#4 (SIR_space_mission_aspect.characterized_object -> considered_aspect
considered_aspect = SIR_kinematic_articulation
SIR_kinematic_articulation <= pair_actuator
pair_actuator.name)

ATTRIBUTE description
#1 (if it is thermal_environment or a material_property_environment)
#2 (if it is a main_body_placement)
#3 (if it is an orbit_arc)
#4 (if it is a kinematic_articulation)
IS_MAPPED_TO
#1 (data_environment.description) USED_FROM 45
#2 (functionally_defined_transformation.description) USED_FROM 43
#3 (characterized_object.name) USED_FROM 41
#4 (SIR_kinematic_articulation.description) USED_FROM SIR
WITH_REFERENCE_PATH
#1 (SIR_space_mission_aspect.characterized_object -> considered_aspect
considered_aspect = data_environment
data_environment.description)
#2 (SIR_space_mission_aspect.characterized_object -> considered_aspect
considered_aspect = SIR_placement_transformation
SIR_placement_transformation <= functionally_defined_transformation
functionally_defined_transformation.description)

#3 (NO MAPPING)
#4 (SIR_space_mission_aspect.characterized_object -> considered_aspect
considered_aspect = SIR_kinematic_articulation
SIR_kinematic_articulation.description)

5.1.7.5 ARM_OBJECT space_mission_event

IS_MAPPED_TO
SIR_space_mission_event USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
SIR_space_mission_event.name USED_FROM SIR

ATTRIBUTE id
IS_MAPPED_TO
SIR_space_mission_event.id USED_FROM SIR

ATTRIBUTE description
IS_MAPPED_TO
SIR_space_mission_event.description USED_FROM SIR

space_mission_event to space_mission_case (as of_case)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_space_mission_event.of_case -> SIR_space_mission_case

5.1.7.6 ARM_OBJECT space_mission_event_with_epoch

IS_MAPPED_TO
SIR_space_mission_event USED_FROM SIR

ATTRIBUTE epoch
IS_MAPPED_TO
date_and_time USED_FROM 41
WITH_REFERENCE_PATH
SIR_space_mission_event.characterization -> event
event = date_and_time

5.1.7.7 ARM_OBJECT space_mission_event_with_running_time

IS_MAPPED_TO
SIR_space_mission_event USED_FROM SIR

ATTRIBUTE running_time
IS_MAPPED_TO
time_measure USED_FROM 41
WITH_REFERENCE_PATH
SIR_space_mission_event.characterization -> event
event = time_measure

5.1.7.8 ARM_OBJECT space_mission_event_with_true_anomaly

IS_MAPPED_TO
SIR_space_mission_event

ATTRIBUTE true_anomaly
IS_MAPPED_TO
true_anomaly USED_FROM SIR
WITH_REFERENCE_PATH
SIR_space_mission_event.characterization -> event
event = true_anomaly

5.1.7.9 ARM_OBJECT space_coordinate_system

IS_MAPPED_TO
TAS_space_coordinate_system USED_FROM TAS

ATTRIBUTE reference_system
IS_MAPPED_TO
representation_context.context_type USED_FROM 43
WITH_REFERENCE_PATH
TAS_space_coordinate_system <= geometric_representation_context
geometric_representation_context <= representation_context
representation_context.context_type

ATTRIBUTE standard_direction
IS_MAPPED_TO
representation_context.context_type USED_FROM 43
WITH_REFERENCE_PATH
TAS_space_coordinate_system <= geometric_representation_context
geometric_representation_context <= representation_context
representation_context.context_type

5.1.7.10 ARM_OBJECT orbit_arc

TAS_orbit_arc USED_FROM TAS

ATTRIBUTE orbit_class
IS_MAPPED_TO
characterized_object.description USED_FROM 41
WITH_REFERENCE_PATH
TAS_orbit_arc <= property_definition_representation
property_definition_representation.definition -> property_definition
property_definition.definition -> characterized_definition
characterized_definition = characterized_object
characterized_object.description

orbit_arc to celestial_body (as governing_celestial_body)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_orbit_arc <= property_definition_representation
property_definition_representation.definition -> property_definition
property_definition <- property_definition_relationship.related_property_definition
{property_definition_relationship => TAS_orbit_generator}
property_definition_relationship.relating_property_definition -> property_definition
{property_definition.description = 'gravitational data'}

orbit_arc to space_mission_event (as evaluation_events)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_orbit_arc <- TAS_evaluation_event.evaluated_items[i]
TAS_evaluation_event.assigned_event -> SIR_space_mission_event

ATTRIBUTE star_spherical_right_ascension
IS_MAPPED_TO
plane_angle_measure_with_unit USED_FROM 41
WITH_REFERENCE_PATH
TAS_orbit_arc <= property_definition_representation
property_definition_representation.used_representation -> representation
{representation => shape_representation}
representation.items[i] -> representation_item
{representation_item.name = 'star spherical right ascension'}
representation_item => measure_representation_item
measure_representation_item <= measure_with_unit
measure_with_unit => plane_angle_measure_with_unit

ATTRIBUTE star_spherical_declination
IS_MAPPED_TO
plane_angle_measure_with_unit USED_FROM 41
WITH_REFERENCE_PATH

TAS_orbit_arc <= property_definition_representation
property_definition_representation.used_representation -> representation
{representation => shape_representation}
representation.items[i] -> representation_item
{representation_item.name = 'star spherical declination'}
representation_item => measure_representation_item
measure_representation_item <= measure_with_unit
measure_with_unit => plane_angle_measure_with_unit

ATTRIBUTE orbit_period
IS_MAPPED_TO
time_measure_with_unit USED_FROM 41
WITH_REFERENCE_PATH
TAS_orbit_arc <= property_definition_representation
property_definition_representation.used_representation -> representation
{representation => shape_representation}
representation.items[i] -> representation_item
representation_item => measure_representation_item
measure_representation_item => TAS_revolution_period

5.1.7.11 ARM_OBJECT discretised_orbit_arc

IS_MAPPED_TO
TAS_discretised_orbit_arc USED_FROM TAS
WITH_REFERENCE_PATH
TAS_discretised_orbit_arc <= TAS_orbit_arc

ATTRIBUTE centre_of_governing_celestial_body
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
TAS_discretised_orbit_arc <= TAS_orbit_arc
TAS_orbit_arc <= property_definition_representation
property_definition_representation.used_representation -> representation
{representation => shape_representation}
representation.items[i] -> representation_item
representation_item => geometric_representation_item
geometric_representation_item => point
point => cartesian_point

discretised_orbit_arc to orbit_position_and_velocity (as positions_and_velocities)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_discretised_orbit_arc <= TAS_orbit_arc
TAS_orbit_arc <= property_definition_representation
property_definition_representation.used_representation -> representation
{representation => shape_representation}
representation.items[i] -> representation_item
representation_item => geometric_representation_item
geometric_representation_item => SIR_position_and_velocity

ATTRIBUTE orbit_generator_name
IS_MAPPED_TO
descriptive_representation_item.description USED_FROM 45
WITH_REFERENCE_PATH
TAS_discretised_orbit_arc <= TAS_orbit_arc
TAS_orbit_arc <= property_definition_representation
property_definition_representation.used_representation -> representation
{representation => shape_representation}
representation.items[i] -> representation_item
{representation_item.name = 'orbit generator'}
representation_item => descriptive_representation_item
descriptive_representation_item.description

discretised_orbit_arc to kepler_parameter_set (as kepler_parameters)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_discretised_orbit_arc <= TAS_orbit_arc
TAS_orbit_arc <= property_definition_representation
property_definition_representation.used_representation -> representation
{representation => shape_representation}
[shape_representation <= representation
representation <- representation_relationship.rep_1
{representation_relationship.description = 'association with unperturbed orbit data'}
representation_relationship.rep_2 -> representation]
[representation.context -> representation_context
representation_context => geometric_representation_context
geometric_representation_context => TAS_space_coordinate_system]
[representation.items[1] -> representation_item
representation_item => SIR_kepler_set]

5.1.7.12 ARM_OBJECT keplerian_orbit_arc

IS_MAPPED_TO
TAS_kepler_orbit_arc USED_FROM TAS
WITH_REFERENCE_PATH
TAS_kepler_orbit_arc <= TAS_orbit_arc

keplerian_orbit_arc to kepler_parameter_set (as kepler_parameters)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_kepler_orbit_arc <= TAS_orbit_arc
TAS_orbit_arc <= property_definition_representation
property_definition_representation.used_representation -> representation
{representation => shape_representation}
representation.items[1] -> representation_item
representation_item => geometric_representation_item
geometric_representation_context => SIR_kepler_set

5.1.7.13 ARM_OBJECT kepler_parameter_set

IS_MAPPED_TO
SIR_kepler_set USED_FROM SIR

ATTRIBUTE semi_major_axis
IS_MAPPED_TO
SIR_kepler_set.semi_major_axis USED_FROM SIR

ATTRIBUTE eccentricity
IS_MAPPED_TO
SIR_kepler_set.eccentricity USED_FROM SIR

ATTRIBUTE inclination
IS_MAPPED_TO
SIR_kepler_set.inclination USED_FROM SIR

ATTRIBUTE right_ascension_of_ascending_node
IS_MAPPED_TO
SIR_kepler_set.right_ascension_of_ascending_node USED_FROM SIR

ATTRIBUTE argument_of_periapsis
IS_MAPPED_TO
SIR_kepler_set.argument_of_periapsis USED_FROM SIR

ATTRIBUTE true_anomaly_at_start
IS_MAPPED_TO
SIR_kepler_set.true_anomaly_at_start USED_FROM SIR

5.1.7.14 ARM_OBJECT orbit_position_and_velocity

IS_MAPPED_TO
SIR_position_and_velocity USED_FROM SIR

orbit_position_and_velocity to cartesian_point (as position)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_position_and_velocity.position -> cartesian_point

ATTRIBUTE velocity_magnitude
IS_MAPPED_TO
measure_with_unit USED_FROM 41
WITH_REFERENCE_PATH
SIR_position_and_velocity.velocity_magnitude -> measure_with_unit
measure_with_unit

ATTRIBUTE velocity_direction
IS_MAPPED_TO
direction USED_FROM 42
WITH_REFERENCE_PATH
SIR_position_and_velocity.velocity_direction -> direction

5.1.7.15 ARM_OBJECT celestial_body

IS_MAPPED_TO
TAS_celestial_body USED_FROM TAS

ATTRIBUTE name
IS_MAPPED_TO
product.name USED_FROM 41
WITH_REFERENCE_PATH
TAS_celestial_body <= product
product.name

ATTRIBUTE radius
#1 (radius is a length_measure)
#2 (radius is a property_function_association)
IS_MAPPED_TO
TAS_radius_of_celestial_body USED_FROM TAS
WITH_REFERENCE_PATH
TAS_celestial_body <= product
product <- product_definition_formation.of_product
product_definition_formation <- product_definition.formation
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition<- property_definition.definition
{property_definition => TAS_gravitational_aspects_of_celestial_body}
property_definition <- property_definition_representation. definition
property_definition_representation.used_representation -> representation
{representation.context_type = ’numerical data’}
representation.items[3] -> representation_item
representation_item => TAS_radius_of_celestial_body
#1 (TAS_radius_of_celestial_body = measure_representation_item
{measure_representation_item <= measure_with_unit
measure_with_unit => length_measure_with_unit})
#2 (TAS_radius_of_celestial_body = SIR_function_valued_property
{SIR_function_valued_property.property_usage.name.name=’length’}
{’SIR_function_valued_property.property_usage.meaning=’radius’})

ATTRIBUTE gravitational_constant
IS_MAPPED_TO
TAS_gravitational_constant USED_FROM TAS
WITH_REFERENCE_PATH
TAS_celestial_body <= product

product <- product_definition_formation.of_product
product_definition_formation <- product_definition.formation
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition<- property_definition.definition
{property_definition => TAS_gravitational_aspects_of_celestial_body}
property_definition <- property_definition_representation. definition
property_definition_representation.used_representation -> representation
{representation.context_type = 'numerical data'}
representation.items[i] -> representation_item
representation_item => TAS_gravitational_constant

ATTRIBUTE sidereal_rotation_period
IS_MAPPED_TO
TAS_revolution_period USED_FROM TAS
WITH_REFERENCE_PATH
TAS_celestial_body <= product
product <- product_definition_formation.of_product
product_definition_formation <- product_definition.formation
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition<- property_definition.definition
{property_definition => TAS_gravitational_aspects_of_celestial_body}
property_definition <- property_definition_representation. definition
property_definition_representation.used_representation -> representation
{representation.context_type = 'numerical data'}
representation.items[i] -> representation_item
representation_item => TAS_sidereal_rotation_period

5.1.7.16 ARM_OBJECT celestial_body_with_orbit

IS_MAPPED_TO
TAS_celestial_body_with_orbit USED_FROM TAS

celestial_body_with_orbit to orbit_arc (as orbit)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_celestial_body_with_orbit <= TAS_celestial_body
TAS_celestial_body <= product
product <- product_definition_formation.of_product
product_definition_formation <- product_definition.formation
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition <- property_definition.definition
property_definition <- TAS_orbit_assignment.related_property_definition
TAS_orbit_assignment.relating_property_definition -> property_definition
property_definition <- property_definition_representation.definition
property_definition_representation => TAS_orbit_arc

5.1.7.17 ARM_OBJECT thermal_aspects_of_celestial_body

IS_MAPPED_TO
TAS_thermal_aspects_of_celestial_body USED_FROM TAS

thermal_aspects_of_celestial_body to celestial_body (as of_celestial_body)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_thermal_aspects_of_celestial_body <= property_definition
property_definition.definition = characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition.formation -> product_definition_formation

product_definition_formation.of_product -> product
product => TAS_celestial_body

ATTRIBUTE black_body_surface_temperature
#1 (black_body_surface_temperature is a thermodynamic_temperature_measure)
#2 (black_body_surface_temperature is a property_function_association)
IS_MAPPED_TO
TAS_black_body_surface_temperature USED_FROM TAS
WITH_REFERENCE_PATH
TAS_thermal_aspects_of_celestial_body <= property_definition
property_definition <- property_definition_representation.definition
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => TAS_black_body_surface_temperature
#1 (TAS_black_body_surface_temperature = measure_representation_item
{measure_representation_item <= measure_with_unit
measure_with_unit => thermodynamic_temperature_measure_with_unit})
#2 (TAS_ black_body_surface_temperature = SIR_function_valued_property
{SIR_function_valued_property.property_usage.name.name='thermodynamic temperature'}
{'SIR_function_valued_property.property_usage.meaning='black body surface temperature'})

ATTRIBUTE notional_distance_to_celestial_body
IS_MAPPED_TO
TAS_notional_distance USED_FROM TAS
WITH_REFERENCE_PATH
TAS_thermal_aspects_of_celestial_body <= property_definition
property_definition <- property_definition_representation.definition
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => TAS_notional_distance

thermal_aspects_of_celestial_body to thermal_radiation_spectrum (as spectrum)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_thermal_aspects_of_celestial_body <= property_definition
property_definition <- property_definition_representation.definition
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => TAS_thermal_radiation_spectrum

thermal_aspects_of_celestial_body to albedo_reflectors (as albedo_reflectors)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_thermal_aspects_of_celestial_body <= property_definition
property_definition.definition = characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition <- product_definition_relationship.related_product_definition
product_definition_relationship => TAS_albedo_reflector

5.1.7.18 ARM_OBJECT albedo_reflector

IS_MAPPED_TO
TAS_albedo_reflector USED_FROM TAS

albedo_reflector to thermal_aspect_of_celestial_body (as albedo_originator)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_albedo_reflector <= product_definition_relationship
product_definition_relationship.relating_product_definition -> product_definition
{product_definition.formation -> product_definition_formation}
product_definition_formation.of_product -> TAS_celestial_body}

ATTRIBUTE albedo_reflection_factor
IS_MAPPED_TO
TAS_albedo_reflection_factor USED_FROM TAS
WITH_REFERENCE_PATH
TAS_albedo_reflector <= product_definition_relationship
product_definition_relationship = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition <- property_definition.definition
{property_definition.description = 'albedo reflection characteristics'}
property_definition <- property_definition_representation.definition
{property_definition_representation => TAS_albedo_reflection_data}
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => TAS_albedo_reflection_factor

albedo_reflector to thermal_radiation_spectrum (as spectrum)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_albedo_reflector <= product_definition_relationship
product_definition_relationship = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition <- property_definition.definition
property_definition <- property_definition_representation.definition
{property_definition_representation => TAS_albedo_reflection_data}
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => SIR_function_valued_property
SIR_function_valued_property => TAS_thermal_radiation_spectrum

5.1.7.19 ARM_OBJECT main_body_placement

IS_MAPPED_TO
SIR_placement_transformation USED_FROM SIR

ATTRIBUTE translation
IS_MAPPED_TO
SIR_placement_transformation.translat USED_FROM SIR

main_body_placement to pointing_in_space (as primary_initial_orientation)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_placement_transformation.primary_orientation -> SIR_pointing_in_space

main_body_placement to pointing_in_space (as secondary_initial_orientation)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_placement_transformation.secondary_orientation -> SIR_pointing_in_space

main_body_placement to attitude_rotation (as rotations)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_placement_transformation.additional_rotations[i] -> rotation_about_direction

5.1.7.20 ARM_OBJECT main_body_placement_with_constant_rotation

IS_MAPPED_TO
TAS_body_motion_representation USED_FROM TAS

ATTRIBUTE rotation_axis
IS_MAPPED_TO
axis1_placement USED_FROM 42
WITH_REFERENCE_PATH

TAS_body_motion <= property_definition_representation
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => SIR_rotation_motion
SIR_rotation_motion.rotation_axis -> axis1_placement

ATTRIBUTE angular_velocity
IS_MAPPED_TO
measure_with_unit USED_FROM 41
WITH_REFERENCE_PATH
TAS_body_motion <= property_definition_representation
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => SIR_rotation_motion
SIR_rotation_motion.rotation_characteristic -> velocity_or_spin_number
velocity_or_spin_number = measure_with_unit

5.1.7.21 ARM_OBJECT main_body_placement_with_spinning

IS_MAPPED_TO
TAS_body_motion_representation USED_FROM TAS

ATTRIBUTE spin_axis
IS_MAPPED_TO
axis1_placement USED_FROM 42
WITH_REFERENCE_PATH
TAS_body_motion <= property_definition_representation
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => SIR_rotation_motion
SIR_rotation_motion.rotation_axis -> axis1_placement

ATTRIBUTE number_of_spin_positions
IS_MAPPED_TO
spin_positions_number USED_FROM SIR
WITH_REFERENCE_PATH
TAS_body_motion <= property_definition_representation
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => SIR_rotation_motion
SIR_rotation_motion.rotation_characteristic -> velocity_or_spin_number
velocity_or_spin_number = spin_positions_number

5.1.7.22 ARM_OBJECT pointing_in_space

IS_MAPPED_TO
SIR_pointing_in_space USED_FROM SIR

ATTRIBUTE pointer
IS_MAPPED_TO
direction USED_FROM 42
WITH_REFERENCE_PATH
SIR_pointing_in_space <= item_defined_transformation
item_defined_transformation.transform_item_1 -> direction

ATTRIBUTE desired_pointing #1 (if desired_pointing is a predefined value) #2 (if desired_pointing is a general_star_pointing) #3 (if desired_pointing is a general_space_pointing)
IS_MAPPED_TO
#1 (descriptive_representation_item.name) USED_FROM 45
#2 (SIR_space_direction) USED_FROM SIR
#3 (direction) USED_FROM 42
WITH_REFERENCE_PATH
SIR_pointing_in_space <= item_defined_transformation
item_defined_transformation.transform_item_2 -> description_or_direction
#1 (description_or_direction = descriptive_representation_item

descriptive_representation_item.name)
#2 (description_or_direction = SIR_space_direction)
#3 (description_or_direction = direction)

5.1.7.23 ARM_OBJECT general_star_pointing

IS_MAPPED_TO
SIR_space_direction

general_star_pointing to space_mission_case (as of_case)
IS_MAPPED_TO
WITH_REFERENCE_PATH
SIR_space_direction = description_or_direction
description_or_direction <- item_defined_transformation.transform_item_2
item_defined_transformation => SIR_pointing_in_space
(SIR_pointing_in_space <- SIR_placement_transformation.primary_orientation)
(SIR_pointing_in_space <- SIR_placement_transformation.secondary_orientation)
SIR_placement_transformation = considered_aspect
considered_aspect <- SIR_space_mission_aspect.characterized_objects[i]
SIR_space_mission_aspect.of_case -> SIR_space_mission_case

ATTRIBUTE name
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
SIR_space_direction <= geometric_representation_item
geometric_representation_item <= representation_item
representation_item.name

ATTRIBUTE star_spherical_right_ascension
IS_MAPPED_TO
SIR_space_direction.star_spherical_right_ascension USED_FROM SIR

ATTRIBUTE star_spherical_declination
IS_MAPPED_TO
SIR_space_direction.star_spherical_declination USED_FROM SIR

5.1.7.24 ARM_OBJECT general_space_pointing

IS_MAPPED_TO
direction USED_FROM 42

general_space_pointing to space_mission_case (as of_case)
IS_MAPPED_TO
WITH_REFERENCE_PATH
SIR_space_direction = description_or_direction
description_or_direction <- item_defined_transformation.transform_item_2
item_defined_transformation => SIR_pointing_in_space
(SIR_pointing_in_space <- SIR_placement_transformation.primary_orientation)
(SIR_pointing_in_space <- SIR_placement_transformation.secondary_orientation)
SIR_placement_transformation = considered_aspect
considered_aspect <- SIR_space_mission_aspect.characterized_objects[i]
SIR_space_mission_aspect.of_case -> SIR_space_mission_case

ATTRIBUTE name
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
direction <= geometric_representation_item
geometric_representation_item <= representation_item
representation_item.name

ATTRIBUTE direction
IS_MAPPED_TO
IDENTICAL MAPPING

5.1.7.25 ARM_OBJECT attitude_rotation

IS_MAPPED_TO
rotation_about_direction USED_FROM 105

ATTRIBUTE axis
IS_MAPPED_TO
rotation_about_direction.direction_of_axis

ATTRIBUTE angle
IS_MAPPED_TO
rotation_about_direction.rotation_angle

5.1.7.26 ARM_OBJECT kinematic_articulation

IS_MAPPED_TO
SIR_kinematic_articulation USED_FROM SIR

kinematic_articulation to kinematic_pair_value (as start_pair_value)
IS_MAPPED_TO
WITH_REFERENCE_PATH
SIR_kinematic_articulation.start_pair_value -> pair_value

kinematic_articulation to kinematic_pair_value (as end_pair_value)
IS_MAPPED_TO
WITH_REFERENCE_PATH
SIR_kinematic_articulation.end_pair_value -> pair_value

5.1.7.27 ARM_OBJECT thermal_environment

IS_MAPPED_TO
TAS_thermal_environment USED_FROM TAS

ATTRIBUTE environmental_sink_name
IS_MAPPED_TO
characterized_object.name USED_FROM 45
WITH_REFERENCE_PATH
TAS_thermal_environment <= data_environment
data_environment.elements[i] -> property_definition_representation
{property_definition_representation => TAS_thermal_environment_data}
property_definition_representation.definition -> characterized_definition
characterized_definition = characterized_object
characterized_object.name

ATTRIBUTE environmental_sink_node_id
IS_MAPPED_TO
NO MAPPING

ATTRIBUTE environmental_sink_temperature
IS_MAPPED_TO
thermodynamic_temperature_measure_with_unit USED_FROM 41
WITH_REFERENCE_PATH
TAS_thermal_environment <= data_environment
data_environment.elements[i] -> property_definition_representation
{property_definition_representation => TAS_thermal_environment_data}
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => measure_representation_item
measure_representation_item <= measure_with_unit
measure_with_unit => thermodynamic_temperature_measure_with_unit

thermal_environment to thermal_aspects_of_celestial_body (as celestial_body_environment)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_thermal_environment <= data_environment
data_environment.elements[i] -> property_definition_representation

property_definition_representation.definition -> property_definition
property_definition => TAS_thermal_aspects_of_celestial_body

ATTRIBUTE star_flux_constant
IS_MAPPED_TO
TAS_thermal_flux USED_FROM TAS
WITH_REFERENCE_PATH
TAS_thermal_environment <= data_environment
data_environment.elements[i] -> property_definition_representation
{property_definition_representation => TAS_thermal_environment_data}
property_definition_representation.used_representation -> representation
representation.items[i] -> representation_item
representation_item => measure_representation_item
measure_representation_item => TAS_thermal_flux

5.1.7.28 ARM_OBJECT thermal_radiation_spectrum

IS_MAPPED_TO
TAS_thermal_radiation_spectrum USED_FROM TAS

ATTRIBUTE name
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
TAS_thermal_radiation_spectrum <= SIR_function_valued_property
SIR_function_valued_property <= representation_item
representation_item.name

ATTRIBUTE description
IS_MAPPED_TO
NO MAPPING

thermal_radiation_spectrum to parameterised_function (as values)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_thermal_radiation_spectrum <= SIR_function_valued_property
SIR_function_valued_property.value

5.1.8 Mapping Table - geometric_model UoF

5.1.8.1 ARM_OBJECT geometric_context

IS_MAPPED_TO
[global_unit_assigned_context] USED_FROM 41
[global_uncertainty_assigned_context] USED_FROM 43
[geometric_representation_context] USED_FROM 42
WITH_RULES 4, 5

geometric_context to length_unit (as length_unit)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
global_unit_assigned_context.units[1] -> unit
unit = named_unit
named_unit => length_unit

geometric_context to plane_angle_unit (as plane_angle_unit)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
global_unit_assigned_context.units[2] -> unit
unit = named_unit
named_unit => plane_angle_unit

ATTRIBUTE point_in_plane_uncertainty
IS_MAPPED_TO

length_measure USED_FROM 41
WITH_REFERENCE_PATH
global_uncertainty_assigned_context.uncertainty[1] -> uncertainty_measure_with_unit
{uncertainty_measure_with_unit.name = 'plane uncertainty'}
uncertainty_measure_with_unit <= measure_with_unit
{measure_with_unit => length_measure_with_unit}
measure_with_unit.value_component -> measure_value
{measure_value = length_measure}

ATTRIBUTE point_coincidence_uncertainty
IS_MAPPED_TO
length_measure USED_FROM 41
IS_MAPPED_TO
global_uncertainty_assigned_context.uncertainty[2] -> uncertainty_measure_with_unit
{uncertainty_measure_with_unit.name = 'point coincidence uncertainty'}
uncertainty_measure_with_unit <= measure_with_unit
{measure_with_unit => length_measure_with_unit}
measure_with_unit.value_component -> measure_value
{measure_value = length_measure}

5.1.8.2 ARM_OBJECT geometric_model

#1 if the model does not contain any boolean construction surface nor any boolean difference surface nor any geometric_submodel_usage with such a characteristic
#2 if the model contains at least one boolean construction surface or one boolean difference surface or a geometric_submodel_usage with such a characteristic
IS_MAPPED_TO
#1 (manifold_surface_shape_representation) USED_FROM 509
#2 (shape_representation) USED_FROM 41

ATTRIBUTE id
IS_MAPPED_TO
representation.name
WITH_REFERENCE_PATH
(manifold_shape_representation <= shape_representation)
shape_representation <= representation
representation.name

ATTRIBUTE name
IS_MAPPED_TO
NO MAPPING

ATTRIBUTE description
#1 if the model does not contain any boolean construction surface nor any boolean difference surface nor any geometric_submodel_usage with such a characteristic
#2 if the model contains at least one boolean construction surface or one boolean difference surface or a geometric_submodel_usage with such a characteristic
IS_MAPPED_TO
descriptive_representation_item.description USED_FROM 45
WITH_REFERENCE_PATH
#1 (NO MAPPING)
#2 (shape_representation <= representation
representation.items[1] -> representation_item
representation_item => descriptive_representation_item
descriptive_representation_item.description)

geometric_model to primitive_bounded_surface (as surfaces)
#1 if the model does not contain any boolean construction surface nor any boolean difference surface nor any geometric_submodel_usage with such a characteristic
#2 if the model contains at least one boolean construction surface or one boolean difference surface or a geometric_submodel_usage with such a characteristic
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
#1 (manifold_shape_representation <= shape_representation

shape_representation <= representation
representation.items[i] -> representation_item
representation => geometric_representation_item
geometric_representation_item => shell_based_surface_model
shell_based_surface_model.sbsm_boundary[i] -> shell
{(shell => open_shell
open_shell <= connected_face_set)
(shell => closed_shell
closed_shell <= connected_face_set)}
connected_face_set.cfs_faces[i] -> face
face => advanced_face)
#2 (shape_representation <= representation
representation.items[i] -> representation_item
representation_item => geometric_representation_item
geometric_representation_item => face_surface
face_surface => advanced_face)

geometric_model to compound_surface_shell (as surfaces)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
(manifold_shape_representation <= shape_representation)
shape_representation <= representation
representation.items[i] -> representation_item
representation_item => topological_representation_item
(topological_representation_item => open_shell)
(topological_representation_item => closed_shell)

geometric_model to boolean_construction_surface (as surfaces)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
shape_representation <= representation
representation.items[i] -> representation_item
representation_item => geometric_representation_item
geometric_representation_item => half_space_solid

geometric_model to boolean_difference_surface (as surfaces)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
shape_representation <= representation
representation.items[i] -> representation_item
representation_item => geometric_representation_item
geometric_representation_item => SIR_boolean_result_surface
WITH_RULES 8

geometric_model to geometric_submodel_usage (as submodel_usages)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
(manifold_shape_representation <= shape_representation)
shape_representation <= representation
representation.items[i] -> representation_item
representation_item => mapped_item

5.1.8.3 ARM_OBJECT geometric_submodel_usage

IS_MAPPED_TO
mapped_item USED_FROM 43

ATTRIBUTE id
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH

mapped_item <= representation_item
representation_item.name

ATTRIBUTE role
IS_MAPPED_TO
NO MAPPING

geometric_submodel_usage to geometric_model (as submodel)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
mapped_item.mapping_source -> representation_map
representation_map.mapped_representation -> representation
representation => shape_representation
(shape_representation => manifold_surface_shape_representation)

geometric_submodel_usage to axis_placement (as placement)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
mapped_item.mapping_target -> representation_item
representation_item => geometric_representation_item
geometric_representation_item => placement
placement => axis2_placement_3d

5.1.8.4 ARM_OBJECT geometric_construction_point

IS_MAPPED_TO
cartesian_point USED_FROM 42

ATTRIBUTE id
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_RULES 3
WITH_REFERENCE_PATH
cartesian_point <= point
point <= geometric_representation_item
geometric_representation_item <= representation_item
representation_item.name

ATTRIBUTE coordinates
IS_MAPPED_TO
cartesian_point.coordinates USED_FROM 42

5.1.8.5 ARM_OBJECT axis_placement

IS_MAPPED_TO
axis2_placement_3d USED_FROM 42

axis_placement to cartesian_point (as location)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
axis2_placement_3d <= placement
placement.location -> cartesian_point

ATTRIBUTE z_axis_direction
IS_MAPPED_TO
axis2_placement_3d.axis USED_FROM 42

ATTRIBUTE x_axis_direction
IS_MAPPED_TO
axis2_placement_3d.ref_direction USED_FROM 42

5.1.8.6 ARM_OBJECT primitive_bounded_surface

IS_MAPPED_TO
advanced_face USED_FROM 511

ATTRIBUTE id
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
advanced_face <= face_surface
face_surface <= face
face <= topological_representation_item
topological_representation_item <= representation_item
representation_item.name

primitive_bounded_surface to axis_placement (as placement)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
advanced_face <= face_surface
face_surface.face_geometry -> surface
surface => surface_replica
surface_replica.transformation -> cartesian_transformation_operator_3d
(see clause 5.2.1 for further details about the transformation)

primitive_bounded_surface to colour_rgb (as colour)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
advanced_face = coloured_item
coloured_item <- TAS_colour_assignment.items[i]
TAS_colour_assignment <= SIR_colour_assignment
SIR_colour_assignment.assigned_colour -> colour
colour => colour_rgb

5.1.8.7 ARM_OBJECT triangle

IS_MAPPED_TO
advanced_face USED_FROM 511
WITH_REFERENCE_PATH
advanced_face <= face_surface
[face_surface.face_geometry -> surface
surface => elementary_surface
elementary_surface => plane]
[face_surface <= face
face.bounds[1] -> face_bound
face_bound => face_outer_bound
face_bound.bound -> loop
loop => edge_loop
edge_loop <= path
path.edge_list[i] -> oriented_edge
oriented_edge.edge_element -> edge]

ATTRIBUTE p1
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]

vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

ATTRIBUTE p2
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

ATTRIBUTE p3
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

5.1.8.8 ARM_OBJECT rectangle

IS_MAPPED_TO
advanced_face USED_FROM 511
WITH_RULES (90 degres)
WITH_REFERENCE_PATH
advanced_face <= face_surface
[face_surface.face_geometry -> surface
surface => elementary_surface
elementary_surface => plane]
[face_surface <= face
face.bounds[1] -> face_bound
face_bound => face_outer_bound
face_bound.bound -> loop
loop => edge_loop
edge_loop <= path
path.edge_list[i] -> oriented_edge
oriented_edge.edge_element -> edge]

ATTRIBUTE p1
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

ATTRIBUTE p2
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

ATTRIBUTE p3
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

5.1.8.9 ARM_OBJECT quadrilateral

IS_MAPPED_TO

advanced_face USED_FROM 511
WITH_REFERENCE_PATH
advanced_face <= face_surface
[face_surface.face_geometry -> surface
surface => elementary_surface
elementary_surface => plane]
[face_surface <= face
face.bounds[1] -> face_bound
face_bound => face_outer_bound
face_bound.bound -> loop
loop => edge_loop
edge_loop <= path
path.edge_list[i] -> oriented_edge
oriented_edge.edge_element -> edge]

ATTRIBUTE p1
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

ATTRIBUTE p2
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

ATTRIBUTE p3
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]

[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

ATTRIBUTE p4
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex]
]
[
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_end -> vertex]
]
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point

5.1.8.10 ARM_OBJECT disc

IS_MAPPED_TO
TAS_disc USED_FROM TAS
WITH_REFERENCE_PATH
TAS_disc <= advanced_face
advanced_face <= face_surface
[face_surface.face_geometry -> surface
surface => swept_surface
swept_surface.swept_curve -> curve
curve => line]
[swept_surface => surface_of_revolution]
[face_surface <= face
face.bounds[1] -> face_bound
face_bound => face_outer_bound
face_bound.bound -> loop
loop => edge_loop
edge_loop <= path
path.edge_list[i] -> oriented_edge
oriented_edge.edge_element -> edge]

ATTRIBUTE p1
IS_MAPPED_TO
cartesian_point USED_FROM 42
WITH_REFERENCE_PATH
[surface_of_revolution.axis_position -> axis1_placement
axis1_placement <= placement
placement.location -> cartesian_point]
[line.pnt -> cartesian_point]

ATTRIBUTE p2
IS_MAPPED_TO

direction USED_FROM 42
WITH_REFERENCE_PATH
surface_of_revolution.axis_position -> axis1_placement
axis1_placement.axis -> direction

ATTRIBUTE external_diameter
IS_MAPPED_TO
positive_length_measure USED_FROM 41
WITH_REFERENCE_PATH
edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle
circle.radius -> positive_length_measure

ATTRIBUTE internal_diameter
IS_MAPPED_TO
positive_length_measure USED_FROM 41
WITH_REFERENCE_PATH
edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle
circle.radius -> positive_length_measure

ATTRIBUTE p3
IS_MAPPED_TO
(mapped indirectly when defining the swept line)

ATTRIBUTE start_angle
IS_MAPPED_TO
[line] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

ATTRIBUTE end_angle
IS_MAPPED_TO
[line] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

5.1.8.11 ARM_OBJECT cylinder

IS_MAPPED_TO
advanced_face USED_FROM 511
WITH_REFERENCE_PATH
advanced_face <= face_surface
[face_surface.face_geometry -> surface
surface => elementary_surface

elementary_surface => cylindrical_surface]
[face_surface <= face
face.bounds[1] -> face_bound
face_bound => face_outer_bound
face_bound.bound -> loop
loop => edge_loop
edge_loop <= path
path.edge_list[i] -> oriented_edge
oriented_edge.edge_element -> edge]

ATTRIBUTE p1
IS_MAPPED_TO
[axis2_placement_3d] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
elementary_surface.position -> axis2_placement_3d
axis2_placement_3d <= placement
placement.location -> cartesian_point

ATTRIBUTE p2
IS_MAPPED_TO
direction USED_FROM 42
WITH_REFERENCE_PATH
axis2_placement_3d.axis -> direction

ATTRIBUTE diameter
IS_MAPPED_TO
[positive_length_measure] USED_FROM 41
[circle] USED_FROM 42
WITH_REFERENCE_PATH
[cylindrical_surface.radius -> positive_length_measure]
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle
circle.radius -> positive_length_measure]

ATTRIBUTE p3
IS_MAPPED_TO
direction USED_FROM 42
WITH_REFERENCE_PATH
axis2_placement_3d.ref_direction -> direction

ATTRIBUTE start_angle
IS_MAPPED_TO
[line] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

ATTRIBUTE end_angle
IS_MAPPED_TO
[line] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]

[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

5.1.8.12 ARM_OBJECT cone

IS_MAPPED_TO
advanced_face USED_FROM 511
WITH_REFERENCE_PATH
advanced_face <= face_surface
[face_surface.face_geometry -> surface
surface => elementary_surface
elementary_surface => conical_surface]
[face_surface <= face
face.bounds[1] -> face_bound
face_bound => face_outer_bound
face_bound.bound -> loop
loop => edge_loop
edge_loop <= path
path.edge_list[i] -> oriented_edge
oriented_edge.edge_element -> edge]

ATTRIBUTE p1
IS_MAPPED_TO
[axis2_placement_3d] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
elementary_surface.position -> axis2_placement_3d
axis2_placement_3d <= placement
placement.location -> cartesian_point

ATTRIBUTE p2
IS_MAPPED_TO
direction USED_FROM 42
WITH_REFERENCE_PATH
axis2_placement_3d.axis -> direction

ATTRIBUTE base_diameter
IS_MAPPED_TO
positive_length_measure USED_FROM 41
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle
circle.radius -> positive_length_measure]
[conical_surface.radius -> length_measure
length_measure => positive_length_measure]

ATTRIBUTE apex_diameter
IS_MAPPED_TO
[positive_length_measure] USED_FROM 41
[plane_angle_measure] USED_FROM 41
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle
circle.radius -> positive_length_measure]
[conical_surface.semi_angle -> plane_angle_measure]

ATTRIBUTE p3
IS_MAPPED_TO

direction USED_FROM 42
WITH_REFERENCE_PATH
axis2_placement_3d.ref_direction -> direction

ATTRIBUTE start_angle
IS_MAPPED_TO
[line] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

ATTRIBUTE end_angle
IS_MAPPED_TO
[line] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => line]
[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

5.1.8.13 ARM_OBJECT sphere

IS_MAPPED_TO
advanced_face USED_FROM 511
WITH_REFERENCE_PATH
advanced_face <= face_surface
[face_surface.face_geometry -> surface
surface => elementary_surface
elementary_surface => spherical_surface]
[face_surface <= face
face.bounds[1] -> face_bound
face_bound => face_outer_bound
face_bound.bound -> loop
loop => edge_loop
edge_loop <= path
path.edge_list[i] -> oriented_edge
oriented_edge.edge_element -> edge]

ATTRIBUTE p1
IS_MAPPED_TO
[axis2_placement_3d] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
elementary_surface.position -> axis2_placement_3d
axis2_placement_3d <= placement
placement.location -> cartesian_point

ATTRIBUTE p2
IS_MAPPED_TO
direction USED_FROM 42
WITH_REFERENCE_PATH
axis2_placement_3d.axis -> direction

ATTRIBUTE diameter
IS_MAPPED_TO
positive_length_measure USED_FROM 41
WITH_REFERENCE_PATH
spherical_surface.radius -> positive_length_measure

ATTRIBUTE base_truncation
IS_MAPPED_TO
[circle] USED_FROM 42
WITH_REFERENCE_PATH
edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle

ATTRIBUTE apex_truncation
IS_MAPPED_TO
[circle] USED_FROM 42
WITH_REFERENCE_PATH
edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle

ATTRIBUTE p3
IS_MAPPED_TO
direction USED_FROM 42
WITH_REFERENCE_PATH
axis2_placement_3d.ref_direction -> direction

ATTRIBUTE start_angle
IS_MAPPED_TO
[circle] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle]
[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

ATTRIBUTE end_angle
IS_MAPPED_TO
[circle] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle]
[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

5.1.8.14 ARM_OBJECT paraboloid

IS_MAPPED_TO
advanced_face USED_FROM 511
WITH_REFERENCE_PATH

advanced_face <= face_surface
[face_surface.face_geometry -> surface
surface => swept_surface
swept_surface.swept_curve -> curve
curve => conic
conic => parabola
swept_surface => surface_of_revolution]
[face_surface <= face
face.bounds[1] -> face_bound
face_bound => face_outer_bound
face_bound.bound -> loop
loop => edge_loop
edge_loop <= path
path.edge_list[i] -> oriented_edge
oriented_edge.edge_element -> edge]

ATTRIBUTE p1
IS_MAPPED_TO
[axis2_placement_3d] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
elementary_surface.position -> axis2_placement_3d
axis2_placement_3d <= placement
placement.location -> cartesian_point

ATTRIBUTE p2
IS_MAPPED_TO
direction USED_FROM 42
WITH_REFERENCE_PATH
axis2_placement_3d.axis -> direction

ATTRIBUTE diameter
IS_MAPPED_TO
length_measure USED_FROM 41
WITH_REFERENCE_PATH
parabola.focal_dist -> length_measure

ATTRIBUTE base_truncation
IS_MAPPED_TO
circle USED_FROM 42
WITH_REFERENCE_PATH
edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle

ATTRIBUTE apex_truncation
IS_MAPPED_TO
circle USED_FROM 42
WITH_REFERENCE_PATH
edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => circle

ATTRIBUTE p3
IS_MAPPED_TO
direction USED_FROM 42
WITH_REFERENCE_PATH
axis2_placement_3d.ref_direction -> direction

ATTRIBUTE start_angle
IS_MAPPED_TO
[parabola] USED_FROM 42
[cartesian_point] USED_FROM 42

WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => parabola]
[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

ATTRIBUTE end_angle
IS_MAPPED_TO
[parabola] USED_FROM 42
[cartesian_point] USED_FROM 42
WITH_REFERENCE_PATH
[edge => edge_curve
edge_curve.edge_geometry -> curve
curve => conic
conic => parabola]
[edge.edge_start -> vertex
edge.edge_end -> vertex
vertex => vertex_point
vertex_point.vertex_geometry -> point
point => cartesian_point]

5.1.8.15 ARM_OBJECT compound_surface_shell

IS_MAPPED_TO
(open_shell) USED_FROM 42
(closed_shell) USED_FROM 42

ATTRIBUTE id
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
(open_shell <= connected_face_set)
(closed_shell <= connected_face_set)
connected_face_set <= topologial_representation_item
topologial_representation_item <= representation_item
representation_item.name

ATTRIBUTE name
IS_MAPPED_TO

compound_surface_shell to primitive_bounded_surface (as surfaces)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
(open_shell <= connected_face_set)
(closed_shell <= connected_face_set)
connected_face_set.cfs_faces -> face
face => face_surface
face_surface => advanced_face

compound_surface_shell to compound_surface_shell (as surfaces)
IS_MAPPED_TO
NO MAPPING (due to the requirement for AIC usage)

5.1.8.16 ARM_OBJECT boolean_construction_surface

IS_MAPPED_TO
half_space_solid USED_FROM 42

ATTRIBUTE id
IS_MAPPED_TO

representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
half_space_solid <= geometric_representation_item
geometric_representation_item <= representation_item
representation_item.name

ATTRIBUTE sense
IS_MAPPED_TO
half_space_solid.agreement_flag USED_FROM 42

boolean_construction_surface to axis_placement (as placement)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
half_space_solid.base_surface -> surface
surface => elementary_surface
elementary_surface.position -> axis2_placement_3d

5.1.8.17 ARM_OBJECT half_space_plane

IS_MAPPED_TO
half_space_solid USED_FROM 42
WITH_REFERENCE_PATH
half_space_solid.base_surface -> surface
{surface => elementary_surface
elementary_surface => plane}

5.1.8.18 ARM_OBJECT half_space_cylinder

IS_MAPPED_TO
half_space_solid USED_FROM 42

ATTRIBUTE radius
IS_MAPPED_TO
cylindrical_surface.radius USED_FROM 42
WITH_REFERENCE_PATH
half_space_solid.base_surface -> surface
{surface => elementary_surface
elementary_surface => cylindrical_surface}
cylindrical_surface.radius

5.1.8.19 ARM_OBJECT generalized_half_space

IS_MAPPED_TO
half_space_solid USED_FROM 42

generalized_half_space to primitive_bounded_surface (as base_surface)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
half_space_solid.base_surface -> surface
surface <- face_surface.face_geometry
{face_surface.same_sense = true}
face_surface => advanced_face

ATTRIBUTE sense
IS_MAPPED_TO
half_space_solid.agreement_flag USED_FROM 42

5.1.8.20 ARM_OBJECT boolean_result_surface

#1 if placement is not specified
#2 if placement is specified
IS_MAPPED_TO
#1 (SIR_boolean_result_surface) USED_FROM SIR
#2 (SIR_boolean_result_replica) USED_FROM SIR

ATTRIBUTE id
IS_MAPPED_TO
representation_item.name USED_FROM 43
WITH_REFERENCE_PATH
#1 (SIR_boolean_result_surface <= geometric_representation_item)
#2 (SIR_boolean_result_replica <= geometric_representation_item)
geometric_representation_item <= representation_item
representation_item.name

boolean_result_surface to axis_placement (as placement)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_boolean_result_replica.transformation -> cartesian_transformation_operator_3d
(see clause 5.2.1 for further details about the transformation)

5.1.8.21 ARM_OBJECT boolean_difference_surface

IS_MAPPED_TO
TAS_boolean_difference_surface USED_FROM TAS
WITH_REFERENCE_PATH
TAS_boolean_difference_surface <= SIR_boolean_result_surface

boolean_difference_surface to primitive_bounded_surface (as first_operand)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_boolean_difference_surface <= SIR_boolean_result_surface
SIR_boolean_result_surface.first_operand -> SIR_boolean_operand_1
SIR_boolean_operand_1 = advanced_face

boolean_difference_surface to compound_bounded_surface (as first_operand)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_boolean_difference_surface <= SIR_boolean_result_surface
SIR_boolean_result_surface.first_operand -> SIR_boolean_operand_1
(SIR_boolean_operand_1 = open_shell)
(SIR_boolean_operand_1 = closed_shell)

boolean_difference_surface to boolean_difference_surface (as first_operand)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_boolean_difference_surface <= SIR_boolean_result_surface
SIR_boolean_result_surface.first_operand -> SIR_boolean_operand_1
SIR_boolean_operand_1 = SIR_boolean_result_surface
{SIR_boolean_result_surface => TAS_boolean_difference_surface}

boolean_difference_surface to boolean_construction_surface (as second_operand)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_boolean_difference_surface <= SIR_boolean_result_surface
SIR_boolean_result_surface.second_operand -> half_space_solid

boolean_difference_surface to generalized_half_space (as second_operand)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_boolean_difference_surface <= SIR_boolean_result_surface
SIR_boolean_result_surface.second_operand -> half_space_solid

5.1.9 Mapping Table - kinematic_model UoF

5.1.9.1 ARM_OBJECT kinematic_model

IS_MAPPED_TO
mechanism USED_FROM 105
WITH_RULES 43

ATTRIBUTE id
IS_MAPPED_TO
NO MAPPING

ATTRIBUTE name
IS_MAPPED_TO
WITH_REFERENCE_PATH
mechanism = named_item
named_item <- TAS_name_assignment.named_items[i]
TAS_name_assignment <= name_assignment
name_assignment.assigned_name

ATTRIBUTE description
IS_MAPPED_TO
NO MAPPING

kinematic_model to kinematic_joint (as joints)
IS_MAPPED_TO
WITH_REFERENCE_PATH
mechanism.structure_definition -> kinematic_structure
kinematic_structure.joints -> kinematic_joint

kinematic_model to kinematic_link (as base)
IS_MAPPED_TO
WITH_REFERENCE_PATH
mechanism.base -> kinematic_link

5.1.9.2 ARM_OBJECT kinematic_link

IS_MAPPED_TO
SIR_kinematic_link USED_FROM SIR
WITH_RULES 42
WITH_REFERENCE_PATH
SIR_kinematic_link <= kinematic_link

kinematic_link to geometric_model (as shape_representation)
IS_MAPPED_TO
WITH_RULES 40
WITH_REFERENCE_PATH
SIR_kinematic_link.klrr -> kinematic_link_representation_relation
kinematic_link_representation_relation.geometrical_aspects -> kinematic_link_representation
kinematic_link_representation <- kinematic_link_representation_association.rep_1
kinematic_link_representation_association <= representation_relationship
representation_relationship.rep_2 -> representation
representation => shape_representation

kinematic_link to product_definition (as associated_product)
IS_MAPPED_TO
WITH_RULES 41
WITH_REFERENCE_PATH
SIR_kinematic_link.represented_product -> characterized_product_definition
characterized_product_definition = product_definition

kinematic_link to product_next_assembly_usage_relationship (as associated_product)
IS_MAPPED_TO
WITH_RULES 41
WITH_REFERENCE_PATH
SIR_kinematic_link.represented_product -> characterized_product_definition
characterized_product_definition = product_definition_relationship

{product_definition_relationship => product_definition_usage
product_definition_usage => assembly_component_usage
assembly_component_usage => next_assembly_usage_occurrence}

5.1.9.3 ARM_OBJECT kinematic_joint

IS_MAPPED_TO
kinematic_joint USED_FROM 105

kinematic_joint to kinematic_link (as first_link)
IS_MAPPED_TO
WITH_REFERENCE_PATH
kinematic_joint.first_link -> kinematic_link
kinematic_link => SIR_kinematic_link

kinematic_joint to kinematic_link (as second_link)
IS_MAPPED_TO
WITH_REFERENCE_PATH
kinematic_joint.second_link -> kinematic_link
kinematic_link => SIR_kinematic_link

5.1.9.4 ARM_OBJECT kinematic_pair

IS_MAPPED_TO
kinematic_pair USED_FROM 105

ATTRIBUTE id
IS_MAPPED_TO
item_defined_transformation.name USED_FROM 43
WITH_REFERENCE_PATH
kinematic_pair <= item_defined_transformation
item_defined_transformation.name

kinematic_pair to kinematic_joint (as joint)
IS_MAPPED_TO
kinematic_pair.joint USED_FROM 105

5.1.9.5 ARM_OBJECT kinematic_pair_value

IS_MAPPED_TO
pair_value USED_FROM 105

ATTRIBUTE applies_to_pair
IS_MAPPED_TO
pair_value.applies_to_pair USED_FROM 105

5.1.9.6 ARM_OBJECT revolute_kinematic_pair

IS_MAPPED_TO
[revolute_pair] USED_FROM 105
[revolute_pair_range] USED_FROM 105
WITH_REFERENCE_PATH
revolute_pair_range <= simple_pair_range
simple_pair_range.applies_to_pair -> revolute_pair

ATTRIBUTE rotation_axis
IS_MAPPED_TO
axis2_placement_3d.axis USED_FROM 42
WITH_REFERENCE_PATH
revolute_pair <= kinematic_pair
kinematic_pair.pair_placement_in_first_link_context -> rigid_placement
rigid_placement=axis2_placement_3d
axis2_placement_3d.axis

ATTRIBUTE lower_limit_actual_rotation
IS_MAPPED_TO
revolute_pair_range.lower_limit_actual_rotation USED_FROM 105

ATTRIBUTE upper_limit_actual_rotation
IS_MAPPED_TO
revolute_pair_range.upper_limit_actual_rotation USED_FROM 105

5.1.9.7 ARM_OBJECT explicit_revolute_kinematic_pair_value

IS_MAPPED_TO
revolute_pair_value USED_FROM 105

ATTRIBUTE actual_rotation
IS_MAPPED_TO
revolute_pair_value.actual_rotation USED_FROM 105

5.1.9.8 ARM_OBJECT implicit_revolute_kinematic_pair_value

IS_MAPPED_TO
SIR_implicit_revolute_pair_value USED_FROM SIR

implicit_revolute_kinematic_pair_value to pointing_in_space (as pointing)
IS_MAPPED_TO
WITH_REFERENCE_PATH
SIR_implicit_revolute_pair_value.pointing -> SIR_pointing_in_space

ATTRIBUTE may_modify_supermodel_pointing_of_first_link
IS_MAPPED_TO
SIR_implicit_revolute_pair_value.impact_on_first_link USED_FROM SIR

5.1.9.9 ARM_OBJECT prismatic_kinematic_pair

IS_MAPPED_TO
[prismatic_pair] USED_FROM 105
[prismatic_pair_range] USED_FROM 105
WITH_REFERENCE_PATH
prismatic_pair_range <= simple_pair_range
simple_pair_range.applies_to_pair -> kinematic_pair
kinematic_pair => prismatic_pair

ATTRIBUTE sliding_axis
IS_MAPPED_TO
axis2_placement_3d.axis USED_FROM 42
WITH_REFERENCE_PATH
prismatic_pair <= kinematic_pair
kinematic_pair.pair_placement_in_first_link_context -> rigid_placement
rigid_placement=axis2_placement_3d
axis2_placement_3d.axis

ATTRIBUTE lower_limit_actual_translation
IS_MAPPED_TO
prismatic_pair_range.lower_limit_actual_translation USED_FROM 105

ATTRIBUTE upper_limit_actual_translation
IS_MAPPED_TO
prismatic_pair_range.upper_limit_actual_translation USED_FROM 105

5.1.9.10 ARM_OBJECT prismatic_kinematic_pair_value

IS_MAPPED_TO
prismatic_pair_value USED_FROM 105

ATTRIBUTE actual_translation
IS_MAPPED_TO
prismatic_pair_value.actual_translation USED_FROM 105

5.1.10 Mapping Table - thermal_radiative_model UoF

5.1.10.1 ARM_OBJECT basic_thermal_radiative_model

IS_MAPPED_TO

TAS_basic_thermal_radiative_model USED_FROM TAS
WITH_REFERENCE_PATH
TAS_basic_thermal_radiative_model <= SIR_model

5.1.10.2 ARM_OBJECT thermal_radiative_face_set

IS_MAPPED_TO
SIR_thermal_radiative_face_set USED_FROM SIR
WITH_RULES 1

thermal_radiative_face_set to primitive_bounded_surface (as surface)
IS_MAPPED_TO
PATH
WITH_RULES 2
WITH_REFERENCE_PATH
SIR_thermal_radiative_face_set.base_surface -> advanced_face

ATTRIBUTE active_side
IS_MAPPED_TO
SIR_thermal_radiative_face_set.active_side USED_FROM TAS

thermal_radiative_face_set to material (as material)
IS_MAPPED_TO
NO MAPPING

5.1.10.3 ARM_OBJECT meshed_thermal_radiative_model

IS_MAPPED_TO
TAS_meshed_thermal_radiative_model USED_FROM TAS
WITH_REFERENCE_PATH
TAS_meshed_thermal_radiative_model <= SIR_model

5.1.10.4 ARM_OBJECT thermal_radiative_meshing

IS_MAPPED_TO
SIR_thermal_radiative_meshing USED_FROM SIR

thermal_radiative_meshing to thermal_radiative_face_set (as face_set)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_thermal_radiative_meshing.base_face -> SIR_thermal_radiative_face_set

ATTRIBUTE meshings_1
IS_MAPPED_TO
SIR_thermal_radiative_meshing.meshing_1 USED_FROM SIR

ATTRIBUTE meshings_2
IS_MAPPED_TO
SIR_thermal_radiative_meshing.meshing_2 USED_FROM SIR

ATTRIBUTE numbering_option
IS_MAPPED_TO
SIR_thermal_radiative_meshing.numbering_option USED_FROM SIR

ATTRIBUTE grid_spacings
IS_MAPPED_TO
[SIR_thermal_radiative_meshing.grid_spacing_1] USED_FROM SIR
[SIR_thermal_radiative_meshing.grid_spacing_2] USED_FROM SIR

5.1.10.5 ARM_OBJECT thermal_radiative_meshing_with_incremental_node_list

IS_MAPPED_TO
SIR_thermal_radiative_incremental_meshing USED_FROM SIR

ATTRIBUTE first_thermal_node_id
IS_MAPPED_TO
SIR_thermal_radiative_incremental_meshing.first_target_id USED_FROM SIR

ATTRIBUTE thermal_node_id_increment
IS_MAPPED_TO
SIR_thermal_radiative_incremental_meshing.id_increment USED_FROM SIR

5.1.10.6 ARM_OBJECT thermal_radiative_meshing_with_full_node_list

IS_MAPPED_TO
SIR_thermal_radiative_explicit_meshing USED_FROM SIR

ATTRIBUTE thermal_node_ids
IS_MAPPED_TO
TAS_thermal_radiative_explicit_meshing.target_ids USED_FROM SIR

5.1.10.7 ARM_OBJECT enclosure

IS_MAPPED_TO
TAS_enclosure USED_FROM TAS
WITH_REFERENCE_PATH
TAS_enclosure <= group_assignment

ATTRIBUTE of_model
IS_MAPPED_TO
NO MAPPING

ATTRIBUTE id
IS_MAPPED_TO
group.name USED_FROM 41
WITH_REFERENCE_PATH
TAS_enclosure <= group_assignment
group_assignment.assigned_group -> group
group.name

ATTRIBUTE name
IS_MAPPED_TO
NO MAPPING

ATTRIBUTE face_sets
IS_MAPPED_TO
TAS_enclosure.grouped_items
WITH_REFERENCE_PATH
TAS_enclosure.grouped_items[i] -> TAS_thermal_radiative_face_set

5.1.11 Mapping Table - visual_presentation UoF

5.1.11.1 ARM_OBJECT colour_rgb

IS_MAPPED_TO
colour_rgb USED_FROM 46

ATTRIBUTE red
IS_MAPPED_TO
colour_rgb.red

ATTRIBUTE green
IS_MAPPED_TO
colour_rgb.green

ATTRIBUTE blue
IS_MAPPED_TO
colour_rgb.blue

5.1.12 Mapping Table - material_properties UoF

5.1.12.1 ARM_OBJECT material

IS_MAPPED_TO
material_designation USED_FROM 45

ATTRIBUTE id
IS_MAPPED_TO
NO MAPPING

ATTRIBUTE name
IS_MAPPED_TO
material_designation.name USED_FROM 45

ATTRIBUTE designation
IS_MAPPED_TO
material_designation.name USED_FROM 45

ATTRIBUTE surface_treatment
IS_MAPPED_TO
WITH_REFERENCE_PATH
material_designation <- material_designation_characterization.designation
material_designation_characterization.property -> characterized_material_property
characterized_material_property = material_property_representation
material_property_representation <= property_definition_representation
property_definition_representation.definition -> property_definition
{property_definition.name = 'surface treatment'}
property_definition_representation.used_representation -> representation
{representation.context_of_items -> representation_context
representation_context.context_type = 'qualitative data'}
representation.items[1] -> representation_item
representation_item => descriptive_representation_item
descriptive_representation_item.description

ATTRIBUTE description
IS_MAPPED_TO
WITH_REFERENCE_PATH
material_designation <- material_designation_characterization.designation
material_designation_characterization.property -> characterized_material_property
characterized_material_property = material_property_representation
material_property_representation <= property_definition_representation
property_definition_representation.definition -> property_definition
{property_definition.name = 'material description'}
property_definition_representation.used_representation -> representation
{representation.context_of_items -> representation_context
representation_context.context_type = 'text data'}
representation.items[1] -> representation_item
representation_item => descriptive_representation_item
descriptive_representation_item.description

5.1.12.2 ARM_OBJECT product_material_relationship

IS_MAPPED_TO
material_designation USED_FROM 41

product_material_relationship to product_definition (as of_product)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
material_designation.definitions[i] -> characterized_definition
characterized_definition = shape_definition
shape_definition = shape_aspect
shape_aspect.of_shape -> product_definition_shape
product_definition_shape <= property_definition
property_definition.definition -> characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definiton = product_definition

product_material_relationship to material (as bulk_material)
IS_MAPPED_TO
PATH

WITH_REFERENCE_PATH
{shape_aspect.description = 'shape interior'}
{shape_aspect.product_definitional = FALSE}
shape_aspect =shape_definition
shape_definition = characterized_definition
characterized_definition <- property_definition.definition
{property_definition.name = 'bulk material'}
property_definition => material_property

product_material_relationship to material (as surface_material)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
{shape_aspect.description = 'shape surface'}
{shape_aspect.product_definitional = TRUE}
shape_aspect =shape_definition
shape_definition = characterized_definition
characterized_definition <- property_definition.definition
{property_definition.name = 'surface material'}
property_definition => material_property

5.1.12.3 ARM_OBJECT material_property

IS_MAPPED_TO
TAS_material_property USED_FROM 45

material_property to material (as of_material)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_material_property <= material_property_representation
material_property_representation = characterized_material_property
characterized_material_property <- material_designation_characterization.property
material_designation_characterization.designation -> material_designation

material_property to material_property_environment (as in_environment)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_material_property <= material_property_representation
material_property_representation.dependent_environment -> data_environment
data_environment => TAS_material_property_environment

material_property to scalar_values (as property_value)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
TAS_material_property <= material_property_representation
material_property_representation <= property_definition_representation
property_definition_representation.used_representation -> representation
representation.items[i] = TAS_property_value

5.1.12.4 ARM_OBJECT material_property_environment

IS_MAPPED_TO
TAS_material_property_environment USED_FROM TAS

5.1.12.5 ARM_OBJECT commercial_aspect_of_material

IS_MAPPED_TO
<TAS_supplier> USED_FROM TAS
<TAS_commercial_designation> USED_FROM TAS

commercial_aspects_of_material to material (as of_material)
IS_MAPPED_TO
PATH

WITH_REFERENCE_PATH
< TAS_commercial_designation.items[i] -> supplied_item
supplied_item = material_designation >
< TAS_supplier.items[i] -> supplied_item
supplied_item = material_designation >

ATTRIBUTE commercial_name
IS_MAPPED_TO
name_assignment.assigned_name USED_FROM 41
WITH_REFERENCE_PATH
TAS_commercial_designation <= name_assignment
name_assignment.assigned_name

ATTRIBUTE supplier
IS_MAPPED_TO
organization.name USED_FROM 41
TAS_supplier <= organization_assignment
{organization_assignment.role -> organization_role
organization_role.name = 'supplier'}
organization_assignment.assigned_organization -> organization
organization.name

5.1.13 Mapping Table - additional_physical_quantities_and_measures UoF

5.1.13.1 ARM_OBJECT derived_unit

IS_MAPPED_TO
derived_unit USED_FROM 41

5.1.13.2 ARM_OBJECT conversion_unit

IS_MAPPED_TO
SIR_unit_with_offset USED_FROM SIR

ATTRIBUTE name
IS_MAPPED_TO
conversion_based_unit.name USED_FROM 41
WITH_REFERENCE_PATH
SIR_unit_with_offset => conversion_based_unit
conversion_based_unit.name

ATTRIBUTE factor
IS_MAPPED_TO
conversion_based_unit.conversion_factor USED_FROM 41
WITH_REFERENCE_PATH
SIR_unit_with_offset => conversion_based_unit
conversion_based_unit.conversion_factor

ATTRIBUTE offset
IS_MAPPED_TO
SIR_unit_with_offset.offset USED_FROM SIR

5.1.13.3 ARM_OBJECT function_valued_property

IS_MAPPED_TO
SIR_function_valued_property USED_FROM SIR

function_valued_property to property_class_usage (as valued_property)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_function_valued_property.property_usage -> SIR_property_usage

ATTRIBUTE value
IS_MAPPED_TO
SIR_function_valued_property.value USED_FROM SIR

5.1.13.4 ARM_OBJECT string_valued_property

IS_MAPPED_TO
SIR_string_valued_property USED_FROM SIR

string_valued_property to property_class_usage (as valued_property)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_function_valued_property.property_usage -> SIR_property_usage

ATTRIBUTE value
IS_MAPPED_TO
SIR_string_valued_property.value USED_FROM SIR

5.1.13.5 ARM_OBJECT number_valued_property

IS_MAPPED_TO
SIR_number_valued_property USED_FROM SIR

number_valued_property to property_class_usage (as valued_property)
IS_MAPPED_TO
PATH
WITH_REFERENCE_PATH
SIR_number_valued_property.property_usage -> SIR_property_usage

ATTRIBUTE value
IS_MAPPED_TO
SIR_number_valued_property.value USED_FROM SIR

5.1.14 Global rules for mapping

The following rules are referenced in the preceding mappings:

1: thermal_face_requires_thermal_model
2: at_most_2_faces_per_surface
3: TAS_point_id_unique
4: TAS_dimensionality_is_3
5: TAS_global_units
6: subtype_mandatory_data_environment
7: restrict_mission_aspects
8: restrict_boolean_operator
9: prod_def_context_requires_prod_def
10: product_context_requires_product
11: unique_ids_in_model
12: subtype_mandatory_product_definition_relationship
13: subtype_mandatory_product_definition_formation
14: subtype_mandatory_date
15: subtype_mandatory_action_relationship
16: subtype_mandatory_action
17: model_requires_definition
18: product_requires_version
19: compatible_dimension
20: exclusive_subtypes_action_assignment
21: dependent_instantiable_type_qualifier
22: dependent_instantiable_security_classification_level
23: dependent_instantiable_person_and_organization_role

24: dependent_instantiable_person
25: dependent_instantiable_date_time_role
26: dependent_instantiable_date
27: dependent_instantiable_approval_status
28: dependent_instantiable_approval_role
29: component_sequence_requires_aspect
30: approvals_are_assigned
31: approval_person_organization_requires_person_organization
32: approval_requires_approval_person_organization
33: approval_date_time_requires_date_time
34: approval_requires_approval_date_time
35: application_context_requires_ap_definition
36: at_most_one_security_level
37: at_most_one_modification_date
38: ato_campaign_requires_contact
39: ato_campaign_requires_creation_date
40: kinematic_link_requires_shape
41: restrict_represented_product
42: subtype_mandatory_kinematic_link
43: mechanism_requires_ground_representation

5.2 AIM EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources and the AICs and contains the types, entity specializations, rules, and functions that are specific to this part of ISO 10303. This clause also specifies modifications to the textual material for the constructs that are imported from the AICs and the integrated resources. The text definitions and EXPRESS specifications provided in the integrated resources for constructs used in the AIM may include select list items and subtypes which are not imported into the AIM. Requirements stated in the integrated resources which refer to such items and subtypes apply exclusively to those items which are imported into the AIM.

```
*)

SCHEMA TAS_schema;

USE FROM SIR_product_version_schema;

USE FROM SIR_parameterized_function_schema;

USE FROM SIR_network_model_schema;

USE FROM SIR_property_schema;

USE FROM SIR_ato_campaign_schema;

USE FROM SIR_measure_schema;

USE FROM SIR_space_mission_schema;

USE FROM SIR_boolean_surfaces_schema;

USE FROM SIR_thermal_radiative_meshing_schema;

USE FROM SIR_colour_assignment_schema;

USE FROM SIR_kinematics_schema;

USE FROM application_context_schema(
                    application_context,
                    product_context,
                    product_definition_context,
                    application_protocol_definition
                    );

USE FROM product_definition_schema(
                    product,
                    product_related_product_category,
                    product_definition,
                    product_definition_relationship
                    );

USE FROM product_property_definition_schema(
                    property_definition,
                    characterized_object,
                    shape_aspect
                    );

USE FROM product_property_representation_schema(
                    property_definition_representation,
                    shape_representation
                    );

USE FROM management_resources_schema(
                    security_classification_assignment,
                    name_assignment,
                    date_and_time_assignment,
                    approval_assignment,
```

```
                    person_and_organization_assignment,
                    organization_assignment,
                    action_assignment,
                    group_assignment
                    );

USE FROM document_schema(
                    document,
                    document_type,
                    document_usage_constraint
                    );

USE FROM security_classification_schema(
                    security_classification_level
                    );

USE FROM approval_schema(
                    approval,
                    approval_person_organization,
                    approval_date_time,
                    approval_status,
                    approval_role
                    );

USE FROM action_schema(
                    action,
                    action_method,
                    executed_action,
                    action_resource,
                    action_relationship
                    );

USE FROM person_organization_schema(
                    address,
                    personal_address,
                    organizational_address,
                    organization,
                    person,
                    person_and_organization,
                    person_and_organization_role
                    );

USE FROM date_time_schema(
                    date,
                    date_and_time,
                    calendar_date,
                    coordinated_universal_time_offset,
                    local_time,
                    date_time_role
                    );

USE FROM group_schema(
                    group
                    );

USE FROM measure_schema;

USE FROM representation_schema(
                    global_uncertainty_assigned_context,
                    representation,
                    representation_item,
                    representation_map,
                    representation_relationship,
                    mapped_item);

USE FROM geometry_schema(
                    geometric_representation_context);
```

```
USE FROM aic_manifold_surface;

USE FROM product_structure_schema(
                    next_assembly_usage_occurrence
                    );

USE FROM material_property_definition_schema(
                    property_definition_relationship,
                    material_property,
                    material_designation,
                    material_designation_characterization);

USE FROM material_property_representation_schema(
                    material_property_representation,
                    data_environment
                    );

USE FROM qualified_measure_schema(
                    measure_representation_item
                    );

USE FROM presentation_resource_schema(
                    colour_rgb
                    );

USE FROM kinematic_structure_schema(
                    mechanism,
                    kinematic_ground_representation,
                    kinematic_joint,
                    kinematic_link_representation_association,
                    kinematic_pair,
                    revolute_pair,
                    revolute_pair_range,
                    revolute_pair_value,
                    prismatic_pair,
                    prismatic_pair_range,
                    prismatic_pair_value
                    );

USE FROM ISO_13584_expressions_schema(
                    int_literal,
                    int_numeric_variable,
                    real_numeric_variable,
                    power_expression,
                    plus_expression,
                    mult_expression
                    );

(*
```

Note 12 - The schemas referenced above can be found in the following Parts:

SCHEMA name	*ISO 10303 part*
SIR_product_version_schema	SIR
SIR_parameterized_function_schema	SIR
SIR_network_model_schema	SIR
SIR_property_schema	SIR
SIR_ato_campaign_schema	SIR
SIR_measure_schema	SIR
SIR_space_mission_schema	SIR
SIR_boolean_surfaces_schema	SIR
SIR_thermal_radiative_meshing_schema	SIR

SCHEMA name	*ISO 10303 part*
`SIR_colour_assignment_schema`	SIR
`SIR_kinematics_schema`	SIR
`application_context_schema`	ISO 10303-41
`measure_schema`	ISO 10303-41
`product_property_definition_schema`	ISO 10303-41
`management_resources_schema`	ISO 10303-41
`product_property_representation_schema`	ISO 10303-41
`document_schema`	ISO 10303-41
`product_definition_schema`	ISO 10303-41
`action_schema`	ISO 10303-41
`approval_schema`	ISO 10303-41
`geometry_schema`	ISO 10303-42
`group_schema`	ISO 10303-41
`representation_schema`	ISO 10303-43
`person_organization_schema`	ISO 10303-41
`date_time_schema`	ISO 10303-41
`material_property_definition_schema`	ISO 10303-45
`material_property_representation_schema`	ISO 10303-45
`qualified_measure_schema`	ISO 10303-45
`presentation_resource_schema`	ISO 10303-46
`kinematic_structure_schema`	ISO 10303-105
`aic_manifold_surface`	ISO 10303-509
`ISO_13584_expressions_schema`	ISO 13584-20

5.2.1 Fundamental concepts and choices

This subclause contains normative statements about the way to create valid instances of entities defined in the AIM schema. Other explanations and advices are provided in the informative annex K.

5.2.1.1 Parameters and result of functions

The parameters and the result of a parameterized function are considered in this protocol as variables.

The definition of a variable of a function has several aspects. A variable has a formal identification suitable for the expression of the function (e.g. y=F(x)). It has also a semantics associating it with a quantity.

The identification of a variable shall be mapped as a subtype of **numeric_variable** (defined in ISO 13584-20). The semantics shall be defined, in the context of this protocol, as a reference to a **SIR_property_usage**.

The combination of both aspects shall be transferred using **SIR_variable.**

5.2.1.2 Polynomial functions

The resources used to map the application object have a structure quite different from the structure proposed in the ARM.

A polynomial function shall be mapped as an instance of **SIR_polynomial_function**.

The polynom shall be transferred as an instance of **numeric_expression** where the components of the polynom shall be mapped using

- **plus_expression**, **mult_expression**, **power_expression** for the operations
- subtypes of **literal_number** for the coefficients,

- **SIR_variable** for the formal parameters,
- **comparison_less_equal** to describe the lower bounds,
- **comparison_greater_equal** to describe the upper bounds.

For the bounds, the first operand of the expression shall be a subtype of **literal_number**, the second shall be a **SIR_variable** defined as one of the parameters of the function.

The degree of the polynomial shall appear in a **power_expression** as an **int_literal**.

5.2.1.3 Property classes

This protocol distinguishes the definition of a class of property and its usage in a given context.

For example, *Length* is a class of property; *Radius* is a usage of the class of property *Length* in a context where some axisymmetric geometry is described.

In this protocol, the context of usage of a property class shall be a **SIR_space_mission_case**.

Therefore, a class of property shall be transferred as a subtype of **SIR_property_class** and a usage of a property class shall be transferred as a **SIR_property_usage**.

5.2.1.4 Relating a product to its various properties

Product and the related entities are the root of every STEP schemas; it upgrades the concept of model found in present CAD systems and in present neutral file exchange standards. The concept of Product is required to deal with configuration management and with multi-representation.

In this AIM, three kinds of products are considered:

- the satellites or any of their components,
- the celestial bodies,
- the materials (envisioned as raw materials).

A **product** shall have one or several **SIR_product_version**s. These **SIR_product_version**s gather views of the same version of the product. These views, e.g. mechanical view, thermal view, ..., shall bee defined as instances of the entity **product_definition**. This entity shall be referred to by one or several instances of **property_definition** corresponding to different representations of the product, e.g CAD model, FEA model, ...

5.2.1.5 Celestial bodies

A celestial body shall be transferred as a **product** belonging the product category ‘CELESTIAL BODIES’. This classification shall be done through an instance of **product_related_product_category**.

Two instances of **product_definition** shall be created to deal respectively with the gravitational data and the thermal data.

5.2.1.6 Representations

The representations shall be transferred as instances of **representation** or its subtypes **shape_representation** and **manifold_surface_shape_representation**. The connection with **property_definition** shall be made through **property_definition_representation** or its subtypes **shape_definition_representation** or **material_property_representation**.

In the case of a representation used to transfer the values of physical quantities, the attribute **context_type** of the associated **representation_context** shall have the value ‘numerical data’.

5.2.1.7 Geometric models

If a geometric_model does not contain any boolean_construction_surface nor any boolean_difference_surface nor a geometric_submodel_usage with such a characteristic, it shall be translated as a **manifold_surface_shape_representation**.

Otherwise, it shall be transferred as a **shape_representation**.

It is reminded that a **manifold_surface_shape_representation** shall contain one or more instances of **shell_based_surface_model**s and shall not refer directly through its inherited attribute item to **advanced_face**s. This implies that the surfaces of the geometric_model shall be translated as **advanced_face**s referred to by a **closed_shell** or an **open_shell**, these objects being referred to by a **shell_based_surface_model**.

5.2.1.8 Sub_model occurences

The definition of a geometric sub-model shall be translated in the same manner than the top-level model.

An occurence of a sub-model shall be translated as follows:

- the positioning aspect of the occurence will be processed as a **representation_map**, the mapped representation of which is the **manifold_surface_shape_representation** (resp. the **shape_representation**) associated with the sub_model. This **representation_map** entity shall be referred to by a **mapped_item** entity which becomes an item of the **manifold_surface_shape_representation** (resp. the **shape_representation**) entity associated with the model. The **mapping_origin** attribute of the **representation_map** shall be an **axis2_placement_3d** defining the origin and axis of the sub_model. The **mapping_target** attribute of the **mapped_item** entity shall be an **axis2_placement_3d** defining the position of the **sub_model_occurence** in the model.
- the configuration management aspect of the occurrence is out of the scope of the current release of the protocol. Therefore there will be no transfer of the associated information.

5.2.1.9 Surfaces

Surfaces defining the shape of the satellites will be transferred as manifold surfaces with topology as defined in ISO 10303-509.

Therefore:

- a triangle, rectangle, quadrilateral, disc, cylinder, cone, sphere or paraboloid shall be transferred as an **advanced_face**.
- A compound_surface_shell shall be transferred as a **closed shell** or an **open_shell**.

It must be noticed that each **advanced_face**, even if the underlying surface is closed, has to be bounded by a **loop**, the **edge**s of which are defined by vertices associated with a **cartesian_point**.

The underlying surface of a triangle, a rectangle, a quadrilateral shall be transferred as an instance of **plane**. The geometry of the edges associated with the advanced_face shall be instances of **line**.

The underlying surface of a disc shall be transferred as an instance of **surface_of_revolution**, the swept curve being a **line**. The geometry of the edges associated with the **advanced_face** shall be two instances of **line** and one or two instances of **circle**.

The underlying surface of a cylinder shall be transferred as an instance of **cylindrical_surface**. The geometry of the edges associated with the **advanced_face** shall be two instances of **line** and two instances of **circle**.

The underlying surface of a shall be transferred as an instance of **conical_surface**. The geometry of the edges associated with the **advanced_face** shall be two instances of **line** and one or two instances of **circle**.

The underlying surface of a shall be transferred as an instance of **spherical_surface**. The geometry of the edges associated with the **advanced_face** shall be two to four instances of **circle**.

The underlying surface of a disc or a paraboloid shall be transferred as an instance of **surface_of_revolution**, the swept curve being a **parabola**. The geometry of the edges associated with the **advanced_face** shall be two instances of **parabola** and one or two instances of **circle**.

5.2.1.10 Geometric construction point

It is reminded that it is not allowed to transfer as an item of a **manifold_shape_representation**, a point that is not referred to, indirectly or directly, by an **advanced_face**, a **shell** or an **axis2_placement_3d**.

Therefore, in the case of isolated (i.e. non referred to) geometric construction points, they shall be either not transferred or transferred in a **shape_representation** that shall not be a **manifold_surface_shape_representation**.

5.2.1.11 Placement of a primitive_bounded_surface or of a boolean_result_surface

In the case where, in the application object, the attribute *placement* is specified, a **cartesian_transformation_operator_3d** shall be instantiated to represent the displacement that shall be applied to the surface (resp. to the result of the boolean operation).

The geometric transformation shall be defined as the transformation that maps the coordinate system of the geometric model that contains the object onto the **axis_placement** specified by *placement*.

Therefore, the instance of **cartesian_transformation_operator_3d** shall be such:

- cartesian_transformation_operator_3d.axis3 is assigned the instance corresponding to placement.z_axis_direction;
- cartesian_transformation_operator_3d\cartesian_transformation_operator.local_origin is assigned the instance corresponding to placement.location;
- cartesian_transformation_operator_3d\cartesian_transformation_operator.axis1 is assigned the instance corresponding to placement.x_axis_direction;
- cartesian_transformation_operator_3d\cartesian_transformation_operator.axis2 is not specified ($);
- cartesian_transformation_operator_3d\cartesian_transformation_operator.scale is not specified or is equal to 1.

5.2.1.12 Kinematic_model

In this version of the protocol, a kinematic_model is intended to represent the kinematic structure of a space product. It is positioned in the coordinate system of the main body.

A kinematic_model shall be transferred as a **mechanism**.

It shall be referred as mechanism.containing_property to a **kinematic_property_definition**.

This **kinematic_property_definition** shall be assigned to a **product_definition** of the main body.

Kinematic_property_definition.ground_definition shall refer to a **characterized_object** with name ‘main body coordinate system’.

A **kinematic_ground_representation** shall be associated through a **kinematic_property_representation_relation** with the ground definition. Its context_of_items shall refer to a **TAS_space_coordinate_system**.

mechanism.base shall refer to a **SIR_kinematic_link**. The represented_product of this latter entity shall be the **product_definition** of the main body.

5.2.1.13 Kinematic_link

A kinematic_link shall be transferred as a **SIR_kinematic_link**.

It shall be associated with a **shape_representation** through a **kinematic_link_representation_association**.

It shall refer as represented_product either to a **product_definition** or to a **next_assembly_usage_occurrence**. In the first case, this **product_definition** shall not be referred to a next_assembly_usage_occurrence|product_definition_relationship.related_product_definition.

5.2.1.14 Material

The definition of a material shall be transferred as data associated with a **product**, using **material_designation**.

The commercial name and the name of the supplier shall be transferred respectively through an instance of **TAS_commercial_name** and **TAS_supplier**.

5.2.1.15 Material_properties

They shall be transferred as instances of **TAS_material_property**.

The material properties shall be associated with the definition of a material through instances of **material_property**, **material_designation_characterization** and **material_designation**.

5.2.1.16 Assignment of units and uncertainties of geometric models

Units and uncertainties are assigned to the representation of a shape on a global basis. This shall be translated creating a complex instance of **geometric_representation_context**, **global_unit_assigned_context** and **global_uncertainty_assigned_context**. Such an instance shall be assigned as the context of a **manifold_surface_shape_representation** (resp. a **shape_representation**) .

5.2.1.17 Assignment of units for property classes

The units specified in the context of a space_mission_case for property classes, shall be transferred through instances of **SIR_unit_assignment**.

5.2.2 TAS_schema type definitions

5.2.2.1 security_classified_item

The **security_classified_item** type provides a mechanism to refer to the entities that can have a security classification level associated with.

EXPRESS specification:

```
*)
TYPE security_classified_item = SELECT (
   SIR_model,
   SIR_node,
   SIR_node_relationship,
   SIR_aspect,
   SIR_ato_campaign,
   SIR_ato_case,
   SIR_ato_phase
   );
END_TYPE;
(*
```

5.2.2.2 dated_item

The **dated_item** type provides a mechanism to refer to the entities that can have a **date_and_time** associated with.

EXPRESS specification:

```
*)
TYPE dated_item = SELECT (
   SIR_ato_campaign,
   NRF_run,
   SIR_parameterized_function
```

```
   );
END_TYPE;
(*
```

5.2.2.3 approved_item

The **approved_item** type provides a mechanism to refer to the entities that can have an approval associated with.

EXPRESS specification:

```
*)
TYPE approved_item = SELECT (
   SIR_ato_campaign
   );
END_TYPE;
(*
```

5.2.2.4 sourced_item

The **sourced_item** type provides a mechanism to refer to the entities that are associated with an information defining a contact person.

EXPRESS specification:

```
*)
TYPE sourced_item = SELECT (
   SIR_ato_campaign
   );
END_TYPE;
(*
```

5.2.2.5 named_item

The **named_item** type provides a mechanism to refer to the entities that can have a name associated with, with using **TAS_name_assignment**.

EXPRESS specification:

```
*)
TYPE named_item = SELECT (
   NRF_run,
   NRF_run_sequence,
   mechanism
   );
END_TYPE;
(*
```

5.2.2.6 documented_item

The **documented_item** SELECT type provides a mechanism to identify entities that may be associated with a **document**.

Express specification:

```
*)
TYPE documented_item = SELECT(
   SIR_parameterized_function);
END_TYPE;
(*
```

5.2.2.7 supplied_item

The **supplied_item** SELECT type provides a mechanism to identify entities for which a commercial designation or a supplier identification may be specified.

Express specification:

```
*)
TYPE supplied_item = SELECT(
   material_designation);
END_TYPE;
(*
```

5.2.2.8 coloured_item

The **coloured_item** SELECT type provides a mechanism to identify entities to which a colour may be assign for presentation purposes.

Express specification:

```
*)
TYPE coloured_item = SELECT(
   advanced_face);
END_TYPE;
(*
```

5.2.2.9 TAS_radius_of_celestial_body

The **TAS_radius_of_celestial_body** SELECT type provides a mechanism that enables to specify the radius of a celestial body either either as a **length_measure_with_unit** or as a **SIR_function_valued_property**.

Express specification:

```
*)
TYPE TAS_radius_of_celestial_body = SELECT (
                    measure_representation_item,
                    SIR_function_valued_property);
WHERE
   WR1: ((‘TAS_SCHEMA.MEASURE_REPRESENTATION_ITEM’ IN TYPEOF(SELF)) AND
 (‘TAS_SCHEMA.LENGTH_MEASURE_WITH_UNIT’ IN TYPEOF(SELF))) XOR
 ((‘TAS_SCHEMA.SIR_FUNCTION_VALUED_PROPERTY’ IN TYPEOF(SELF)) AND
 (SELF\SIR_function_valued_property.property_usage.meaning.roles[1].name
=’radius’) AND
 (SELF\SIR_function_valued_property.property_usage.property.name.name=’length’));
END_TYPE;
(*
```

Formal proposition:

WR1: a **TAS_radius_of_celestial_body** shall either be:

- a **measure_representation_item** and simultaneously a **length_measure_with_unit**, or
- a **SIR_function_valued_property** ‘such the valued property has for name ‘length’ and for meaning ‘radius’.

Informal proposition:

IP1: in the case where the **TAS_radius_of_celestial_body** is a **SIR_function_valued_property**, SELF.value shall refer to a **SIR_parameterized_function** that has one or two parameters corresponding to the latitude or/and to the longitude of the considered celestial body.

5.2.2.10 TAS_black_body_surface_temperature

The **TAS_black_body_surface_temperature** SELECT type provides a mechanism that enables to specify the equivalent black body temperature of the surface of a celestial body either as a **thermodynamic_temperature_measure_with_unit** or as a **SIR_function_valued_property**.

Express specification:

```
*)
TYPE TAS_black_body_surface_temperature = SELECT (
                    measure_representation_item,
                    SIR_function_valued_property);
```

```
WHERE
   WR1: (('TAS_SCHEMA.MEASURE_REPRESENTATION_ITEM' IN TYPEOF(SELF)) AND
 ('TAS_SCHEMA.THERMODYNAMIC_TEMPERATURE_MEASURE_WITH_UNIT' IN TYPEOF(SELF))) XOR
 (('TAS_SCHEMA.SIR_FUNCTION_VALUED_PROPERTY' IN TYPEOF(SELF)) AND
 (SELF\SIR_function_valued_property.property_usage.meaning.roles[1].name ='black
body surface temperature') AND

(SELF\SIR_function_valued_property.property_usage.property.name.name='thermodynami
c temperature'));
END_TYPE;
(*
```

Formal proposition:

WR1: a **TAS_black_body_surface_temperature** shall either be:

- a **measure_representation_item** and simultaneously a **thermodynamic_temperature_with_unit**, or
- a **SIR_function_valued_property** 'such the valued property has for name 'thermodynamic temperature' and for meaning 'black body surface temperature'.

Informal proposition:

IP1: in the case where the **TAS_black_body_surface_temperature** is a **SIR_function_valued_property**, SELF.value shall refer to a **SIR_parameterized_function** that has one or two parameters corresponding to the latitude or/and to the longitude of the considered celestial body.

5.2.2.11 TAS_albedo_reflection_factor

The **TAS_albedo_reflection_factor** SELECT type provides a mechanism that enables to specify an albedo reflection factor either as a **ratio_measure_with_unit** or as a **SIR_function_valued_property**.

The constant or the function representing the albedo reflection factor shall be in the range [0.,1.].

Note 13 - The unit of the albedo reflection factor is deduced from the units of the quantities taken into account in the ratio.

Express specification:

```
*)
TYPE TAS_albedo_reflection_factor = SELECT (
                    measure_representation_item,
                    SIR_function_valued_property);
WHERE
   WR1: (('TAS_SCHEMA.MEASURE_REPRESENTATION_ITEM' IN TYPEOF(SELF)) AND
 ('TAS_SCHEMA.RATIO_MEASURE_WITH_UNIT' IN TYPEOF(SELF))) XOR
 (('TAS_SCHEMA.SIR_FUNCTION_VALUED_PROPERTY' IN TYPEOF(SELF)) AND
 (SELF\SIR_function_valued_property.property_usage.meaning.roles[1].name ='albedo
reflection factor') AND
 (SELF\SIR_function_valued_property.property_usage.property.name.name='ratio'));
END_TYPE;
(*
```

Formal proposition:

WR1: a **TAS_albedo_reflection_factor** shall either be:

- a **measure_representation_item** and simultaneously a **ratio_measure_with_unit**, or
- a **SIR_function_valued_property** 'such the valued property has for name 'ratio' and for meaning 'albedo_reflection_factor'.

Informal proposition:

IP1: in the case where the **TAS_albedo_reflection_factor** is a **SIR_function_valued_property**, SELF.value shall refer to a **SIR_parameterized_function** that has one or two parameters corresponding to the latitude or/and to the longitude of the considered celestial body;

IP2: in any case, a **TAS_albedo_reflection_factor** shall be in the range [0.,1.].

5.2.2.12 TAS_property_value

The **TAS_property_value** SELECT type provides a mechanism to refer to the entities that may carry the value of a material property.

Express specification:

```
*)
TYPE TAS_property_value = SELECT(
   SIR_number_valued_property,
   SIR_function_valued_property,
   SIR_string_valued_property
   );
END_TYPE;
(*
```

5.2.3 TAS_schema entity definitions

5.2.3.1 TAS_security_assignment

A **TAS_security_assignment** is a kind of **security_classification_assignment** that enables to assign a security_classification_level to **security_classified_item**s.

The following values are permitted for specifying the security level:

- ‘unclassified’: identifies the classification level for which no security is necessary;
- ‘classified’: identifies the classification level for which security is necessary but the classification details are not given;
- ‘proprietary’: identifies the classification level for which the disclosure of information about the assigned items would risk an organization’s market or competitive advantage ;
- ‘confidential’: identifies the classification level for which the disclosure of information about the assigned items would cause damage to national or organizational security;
- ‘secret’: identifies the classification level for which the disclosure of information about the assigned items would cause serious damage to national or organizational security ;
- ‘top_secret’: identifies the classification level for which the disclosure of information about the assigned items would cause exceptionally grave damage to national or organizational security;

EXPRESS specification:

```
*)
ENTITY TAS_security_assignment
SUBTYPE OF (security_classification_assignment);
   items: SET[1:?] OF security_classified_item;
WHERE
   WR1:
SELF\security_classification_assignment.assigned_security_classification.security_
level.name IN
[‘unclassified’,’classified’,’proprietary’,’confidential’,’secret’,’top_secret’];
END_ENTITY;
(*
```

Attribute definitions:

items: specifies a set of **security_classified_item** to which a security level is assigned.

Formal proposition:

WR1: the name of the **security_level** assigned to items shall be one of the following values: ‘unclassified’,’classified’,’proprietary’,’confidential’,’secret’,’top_secret’.

5.2.3.2 TAS_date_assignment

A **TAS_date_assignment** is a kind of **date_and_time_assignment** that enables to assign a **date_and_time** to **dated_item**s.

EXPRESS specification:

```
*)
ENTITY TAS_date_assignment
SUBTYPE OF (date_and_time_assignment);
   items: SET[1:?] OF dated_item;
END_ENTITY;
(*
```

Attribute definitions:

items: specifies a set of **dated_items** to which a **date_and_time** is assigned.

5.2.3.3 TAS_approval_assignment

A **TAS_approval_assignment** is a kind of **approval_assignment** that enables to assign an **approval** to **approved_item**s.

EXPRESS specification:

```
*)
ENTITY TAS_approval_assignment
SUBTYPE OF (approval_assignment);
   items: SET[1:?] OF approved_item;
END_ENTITY;
(*
```

Attribute definitions:

items: specifies a set of **approved_items** to which an **approval** is assigned.

5.2.3.4 TAS_contact_assignment

A **TAS_contact_assignment** is a kind of **person_and_organization_assignment** that enables to identify a contact for a **sourced_item**s.

EXPRESS specification:

```
*)
ENTITY TAS_contact_assignment
SUBTYPE OF (person_and_organization_assignment);
   items: SET[1:?] OF sourced_item;
WHERE
   WR1:
SELF\person_and_organization_assignment.role.name='contact_person/organization';
END_ENTITY;
(*
```

Attribute definitions:

items: specifies a set of **sourced_items** to which a contact **person_and_organization** is assigned.

Formal proposition:

WR1: the role of the assigned **person_and_organization** shall be 'contact_person/organization'.

5.2.3.5 TAS_name_assignment

A **TAS_name_assignment** is a kind of **name_assignment** that enables to assign a name to **named_item**s.

EXPRESS specification:

```
*)
ENTITY TAS_name_assignment
SUBTYPE OF (name_assignment);
   items: SET[1:?] OF named_item;
END_ENTITY;
(*
```

Attribute definitions:

items: specifies a set of **named_items** to which a name is assigned.

5.2.3.6 TAS_commercial_designation

A **TAS_commercial_designation** is a kind of **name_assignment** that enables to assign a commercial name to **supplied_item**s.

EXPRESS specification:

```
*)
ENTITY TAS_commercial_designation
SUBTYPE OF (name_assignment);
   items: SET[1:?] OF supplied_item;
END_ENTITY;
(*
```

Attribute definitions:

items: specifies a set of **supplied_items** to which a commercial name is assigned.

5.2.3.7 TAS_supplier

A **TAS_supplier** is a kind of **organization_assignment** that identifies an **organization** in its role of supplier of **supplied_item**s.

EXPRESS specification:

```
*)
ENTITY TAS_supplier
SUBTYPE OF (organization_assignment);
   items: SET[1:?] OF supplied_item;
WHERE
   WR1: SELF\organization_assignment.role.name = 'supplier'
END_ENTITY;
(*
```

Attribute definitions:

items: specifies a set of **supplied_items** to which a commercial name is assigned.

Formal proposition:

WR1: the role of the organization with respect to the supplied items shall be 'supplier'.

5.2.3.8 TAS_document_reference

A **TAS_document_reference** is a kind of **document_reference** that enables to associate a **document** with **documented_item**s.

EXPRESS specification:

```
*)
ENTITY TAS_document_reference
SUBTYPE OF (document_reference);
   documented_items: SET[1:?] OF documented_item;
END_ENTITY;
(*
```

Attribute definitions:

documented_items: specifies a set of **documented_item**s a **document** is associated with.

5.2.3.9 TAS_colour_assignment

A **TAS_colour_assignment** is a kind of **SIR_colour_assignment** that assign, for presentation purposes, a **colour_rgb** to a set of **coloured_item**s.

The method for displaying the **coloured_item**s and the other characteristics of their graphical aspect are not specified.

Note 14 - A reading interface is therefore let free as regards the way to deal with this information.

EXPRESS specification:

```
*)
ENTITY TAS_colour_assignment
SUBTYPE OF (SIR_colour_assignment);
   items: SET[1:?] OF coloured_item;
   SELF\SIR_colour_assignment.assigned_colour: colour_rgb;
END_ENTITY;
(*
```

Attribute definitions:

items: specifies a set of **coloured_items** to which a colour is assigned.

SELF\SIR_colour_assignment.assigned_colour: specifies the **colour_rgb** that is assigned;

5.2.3.10 TAS_model_product_relationship

A **TAS_model_product_relationship** is a kind of **property_definition_representation** that enables to associate a **SIR_model** of a constituent of a **SIR_model** with a **product_definition** or a **shape_aspect**, in order to specify that this representation or this element of representation represents a particular product or a portion of the shape of the product.

Note 15 - This entity is similar to the entity **NRF_model_product_relationship** in the protocol STEP-NRF with the main addition that the relationship may be made with a **shape_aspect**. This enables to associate a **SIR_thermal radiative_face_set** with a **shape_aspect**.

EXPRESS specification:

```
*)
ENTITY TAS_model_product_relationship
SUBTYPE OF (property_definition_representation) ;
WHERE
   WR1: SELF\property_definition_representation.definition.description='shape
description for space mission case';
   WR2:SIZEOF(['TAS_SCHEMA.PRODUCT_DEFINITION','TAS_SCHEMA.SHAPE_ASPECT’] *
TYPEOF(SELF\property_definition_representation.definition.definition))=1;
   WR3: ('TAS_SCHEMA.SIR_MODEL' IN
TYPEOF(SELF\property_definition_representation.used_representation)) XOR
((SIZEOF(SELF\property_definition_representation.used_representation.items)=1)
AND
(SIZEOF(['TAS_SCHEMA.SIR_NODE','TAS_SCHEMA.SIR_NODE_RELATIONSHIP','TAS_SCHEMA.SIR_
SUBMODEL_USAGE']*TYPEOF(SELF\property_definition_representation.used_representatio
n.items[1]))=1));
END_ENTITY;
(*
```

Formal propositions:

WR1: the name of the **property_definition** referred to by the **TAS_model_relationship** shall be 'product description for space mission case';

WR2: the **property_definition** referred to by the **TAS_model_relationship** shall be a property of a **product_definition** or of a **shape_aspect**;

WR3: the representation referred to by the **TAS_model_relationship** shall be a **SIR_model** or it shall contain only one element in items and the only element of items shall be of type **SIR_node** or **SIR_node_relationship** or **SIR_submodel_usage**.

5.2.3.11 NRF_run

A **NRF_run** is a kind of **executed_action** made with the data of a **SIR_ato_phase**.

EXPRESS specification:

```
*)
ENTITY NRF_run
SUBTYPE OF (executed_action);
DERIVE
   run_name:  SET OF label := get_names( SELF);
   creator_tool_or_facility: SET OF action_resource:= get_tools( SELF);
   timestamp: SET OF date_and_time := get_timestamps(SELF);
INVERSE
   subject_association: NRF_run_phase FOR assigned_action;
   results_association: NRF_run_results FOR assigned_action;
UNIQUE
   UR1: SELF\action.name, subject_association;
WHERE
   WR1: SIZEOF(run_name) = 1;
   WR2: SIZEOF(creator_tool_or_facility) = 1;
   WR3: SIZEOF(timestamp) = 1;
END_ENTITY;
(*
```

Attribute definitions:

run_name: specifies a set of labels assigned to the **NRF_run**;

creator_tool_or_facility: specifies a set of **action_resource** that refer to the **NRF_run**;

timestamp: specifies a set of **date_and_time** assigned to the **NRF_run**;

subject_association: specifies the **NRF_run_phase** that refers to the **NRF_run**;

results_association: specifies the **NRF_run_results** that refers to the **NRF_run**.

Formal propositions:

UR1: there shall not be more than one **NRF_run** with the same id and subject;

WR1: there shall be exactly one label assigned to the **NRF_run** as run_name;

WR2: there shall be at most one **action_resource** assigned to the **NRF_run**;

WR3: there shall be exactly one **date_and_time** assigned to the **NRF_run**.

5.2.3.12 NRF_run_phase

A **NRF_run_phase** is a kind of **action_assignment** that associates a **NRF_run** with the **SIR_phase** that is the subject of the run.

EXPRESS specification:

```
*)
ENTITY NRF_run_phase
SUBTYPE OF (action_assignment);
   subject: SIR_ato_phase;
   SELF\action_assignment.assigned_action: NRF_run;
END_ENTITY;
(*
```

Attribute definitions:

subject: specifies the **SIR_phase** for which the **NRF_run** has been executed.

SELF\action_assignment.assigned_action: specifies the considered **NRF_run**.

5.2.3.13 NRF_run_results

A **NRF_run_results** is a kind of **action_assignment** that associates a **NRF_run** with the **SIR_aspect** that results from the run.

EXPRESS specification:

```
*)
ENTITY NRF_run_results
SUBTYPE OF (action_assignment);
   results: SET[1:?] OF SIR_aspect;
   SELF\action_assignment.assigned_action: NRF_run;
END_ENTITY;
(*
```

Attribute definitions:

results: specifies the **SIR_aspects** that result from the **NRF_run**.

SELF\action_assignment.assigned_action: specifies the considered **NRF_run**;

Formal propositions:

WR1: the assigned_action shall be a **NRF_run**.

5.2.3.14 NRF_run_sequence

A **NRF_run_sequence** is a kind of **action** that collects a set of one or more **NRF_runs** dealing with **SIR_phases** belonging to the same **SIR_ato_case**.

EXPRESS specification:

```
*)
ENTITY NRF_run_sequence
SUBTYPE OF (action);
DERIVE
   sequence_name:  SET OF label := get_names( SELF);
INVERSE
   subject_association: NRF_run_sequence_case FOR assigned_action;
   sequence: SET[1:?] OF NRF_run_in_sequence FOR relating_action;
UNIQUE
   UR1: SELF\action.name, subject_association;
WHERE
   WR1: SIZEOF(sequence_name) = 1;
END_ENTITY;
(*
```

Attribute definitions:

sequence_name: specifies a set of labels assigned to the **NRF_run**;

subject_association: specifies the **NRF_run_sequence_case** that refers to the **NRF_run_sequence**;

sequence: specifies a set of one or more **NRF_run_in_sequence**s that specify the **NRF_runs** that are included in the **NRF_run_sequence**.

Formal propositions:

UR1: there shall not be more than one **NRF_run_sequence** with the same SELF\action.name and subject;

WR1: there shall be exactly one label assigned to the **NRF_run_sequence** as sequence_name.

5.2.3.15 NRF_run_sequence_case

A **NRF_run_sequence_case** is a kind of **action_assignment** that associates a **NRF_run_sequence** with the **SIR_ato_case** that is the subject of the runs included in the sequence.

EXPRESS specification:

```
*)
ENTITY NRF_run_sequence_case
SUBTYPE OF (action_assignment);
   subject: SIR_ato_case;
   SELF\action_assignment.assigned_action: NRF_run_sequence;
END_ENTITY;
(*
```

Attribute definitions:

subject: specifies the **SIR_ato_case** for which the **NRF_run_sequence** has been executed;

SELF\action_assignment.assigned_action: specifies the considered **NRF_run_sequence**.

5.2.3.16 NRF_run_in_sequence

A **NRF_run_in_sequence** is a kind of **action_relationship** that associates a **NRF_run** with the **NRF_run_sequence** that it is a member of.

EXPRESS specification:

```
*)
ENTITY NRF_run_in_sequence
SUBTYPE OF (action_relationship);
   SELF\action_relationship.relating_action: NRF_run_sequence;
   SELF\action_relationship.related_action: NRF_run;
WHERE
   WR1: related_action.subject_association.subject.of_case :=:
relating_action.subject_association.subject;
END_ENTITY;
(*
```

Attribute definitions:

SELF\action_relationship.relating_action: specifies the considered **NRF_run_sequence**;

SELF\action_relationship.related_action: specifies a **NRF_run** that is a member of the **NRF_run_sequence**.

Formal propositions:

WR1: the **SIR_ato_phase**, subject of the related **NRF_run** shall refer to the **SIR_ato_case** that is the subject of the **NRF_run_sequence**.

5.2.3.17 TAS_space_coordinate_system

A **TAS_space_coordinate_system** entity defines a reference coordinate system used in the definition of orbit trajectories and spacecraft attitudes.

As, it is a subtype of **geometric_representation_context**, the rule **TAS_dimensionality_is_3** also applies to any instance of **TAS_space_coordinate_system**.

EXPRESS specification:

```
*)
ENTITY TAS_space_coordinate_system
SUBTYPE OF (geometric_representation_context);
WHERE
   WR1: (SELF\representation_context.context_type = 'star centric inertial ecliptic with
gamma_50 vernal equinox') OR
(SELF\representation_context.context_type = 'star centric inertial ecliptic with gamma_2000
vernal equinox') OR
(SELF\representation_context.context_type = 'star centric inertial ecliptic with TOD
vernal equinox') OR
(SELF\representation_context.context_type = 'planet centric inertial ecliptic with gamma_50
vernal equinox') OR
```

```
(SELF\representation_context.context_type = 'planet centric inertial ecliptic with
gamma_2000 vernal equinox') OR
(SELF\representation_context.context_type = 'planet centric inertial ecliptic with TOD
vernal equinox') OR
(SELF\representation_context.context_type = 'planet centric inertial equatorial with
gamma_50 vernal equinox') OR
(SELF\representation_context.context_type = 'planet centric inertial equatorial with
gamma_2000 vernal equinox') OR
(SELF\representation_context.context_type = 'planet centric inertial equatorial with TOD
vernal equinox') OR
(SELF\representation_context.context_type = 'planet centric star fixed ecliptic') OR
(SELF\representation_context.context_type = 'planet centric star fixed equatorial' ) OR
(SELF\representation_context.context_type = 'planet centric local');
END_ENTITY;
(*
```

Formal propositions:

WR1: the inherited attribute **context_type** shall be either:

- 'star centric inertial ecliptic with gamma_50 vernal equinox' or
- 'star centric inertial ecliptic with gamma_2000 vernal equinox' or
- 'star centric inertial ecliptic with TOD vernal equinox' or
- 'planet centric inertial ecliptic with gamma_50 vernal equinox' or
- 'planet centric inertial ecliptic with gamma_2000 vernal equinox' or
- 'planet centric inertial ecliptic with TOD vernal equinox' or
- 'planet centric inertial equatorial with gamma_50 vernal equinox' or
- 'planet centric inertial equatorial with gamma_2000 vernal equinox' or
- 'planet centric inertial equatorial with TOD vernal equinox' or
- 'planet centric star fixed ecliptic' or
- 'planet centric star fixed equatorial' or
- 'planet centric local'.

5.2.3.18 TAS_gravitational_constant

A **TAS_gravitational_constant** entity is the result of the product of the Newtonian constant of gravitation with the mass of a celestial body.

Its dimension is $L^3 * T^{-2}$.

EXPRESS specification:

```
*)
ENTITY TAS_gravitational_constant
SUBTYPE OF (measure_representation_item);
DERIVE
   dimens: dimensional_exponents :=
derive_dimensional_exponents(SELF\measure_with_unit.unit_component);
WHERE
   WR1: SELF\measure_with_unit.value_component >0.;
   WR2: dimens.length_exponent =3;
   WR3: dimens.time_exponent=-2;
   WR4: (dimens.mass_exponent=0) AND (dimens.electric_current_exponent=0) AND
(dimens.thermodynamic_temperature_exponent=0) AND
(dimens.amount_of_substance_exponent=0) AND (dimens.luminous_intensity_exponent=0)
;
END_ENTITY;
(*
```

Attribute definitions:

dimens: specifies the exponent of each of the seven fundamental quantities, in the dimensional equation of the unit of the measure;

Formal propositions:

WR1: the value of the measure shall be greater than 0;

WR2: the exponent of length in the dimensional equation shall be equal to 3;

WR3: the exponent of time in the dimensional equation shall be equal to -2;

WR4: the other exponents shall be zero.

5.2.3.19 TAS_revolution_period

A **TAS_revolution_period** is the measure of the time needed to make a complete circular or elliptic revolution .

EXPRESS specification:

```
*)
ENTITY TAS_revolution_period
SUBTYPE OF (measure_representation_item);
WHERE
   WR1: SELF\measure_with_unit.value_component >0.;
   WR2: 'TAS_SCHEMA.TIME_MEASURE_WITH_UNIT' IN TYPEOF(SELF);
END_ENTITY;
(*
```

Formal propositions:

WR1: the value of the measure shall be greater than 0;

WR2: the revolution period shall be expressed as a **time_measure_with_unit**.

5.2.3.20 TAS_notional_distance

A **TAS_notional_distance** is the measure of the distance to be used for calculation of the thermal flux coming from a celestial body.

EXPRESS specification:

```
*)
ENTITY TAS_notional_distance
SUBTYPE OF (measure_representation_item);
WHERE
   WR1: SELF\measure_with_unit.value_component >0.;
   WR2: 'TAS_SCHEMA.LENGTH_MEASURE_WITH_UNIT' IN TYPEOF(SELF);
END_ENTITY;
(*
```

Formal propositions:

WR1: the value of the measure shall be greater than 0;

WR2: the notional distance shall be expressed as a **length_measure_with_unit**.

5.2.3.21 TAS_thermal_flux

A **TAS_thermal_flux** is the measure with unit of a thermal radiation flux.

Its dimensional expression of which being $M.T^{-3}$.

EXPRESS specification:

```
*)
ENTITY TAS_thermal_flux
SUBTYPE OF (measure_representation_item);
WHERE
   wr1: 'TAS_SCHEMA.NAMED_UNIT' IN TYPEOF(SELF\measure_with_unit.unit_component);
   wr2:
(SELF\measure_with_unit.unit_component\named_unit.dimensions.mass_exponent = 1)
AND
```

```
(SELF\measure_with_unit.unit_component\named_unit.dimensions.time_exponent = -3)
AND
(SELF\measure_with_unit.unit_component\named_unit.dimensions.length_exponent = 0)
AND
(SELF\measure_with_unit.unit_component\named_unit.dimensions.electric_current_expo
nent = 0) AND
(SELF\measure_with_unit.unit_component\named_unit.dimensions.thermodynamic_tempera
ture_exponent = 0) AND
(SELF\measure_with_unit.unit_component\named_unit.dimensions.amount_of_substance_e
xponent = 0) AND
(SELF\measure_with_unit.unit_component\named_unit.dimensions.luminous_intensity_ex
ponent = 0);
END_ENTITY;
(*
```

Formal propositions:

wr1: the unit of a **TAS_thermal_flux** shall be a **named_unit**;

wr2: its dimensional exponents shall be MT^{-3};

5.2.3.22 TAS_thermal_radiation_spectrum

A **TAS_thermal_radiation_spectrum** entity specifies the intensity of thermal electromagnetic radiation as a function of wavelength.

It is a subtype of **SIR_function_valued_property**.

EXPRESS specification:

```
*)
ENTITY TAS_thermal_radiation_spectrum
SUBTYPE OF (SIR_function_valued_property);
END_ENTITY;
(*
```

Informal propositions :

IP1: the radiation spectrum shall be defined as a function of wavelength.

5.2.3.23 TAS_evaluation_event

A **TAS_evaluation_event** entity identifies a **SIR_space_mission_event** that is an evaluation point (or instant) for one or more **TAS_orbit_arc**s.

EXPRESS specification:

```
*)
ENTITY TAS_evaluation_event
SUBTYPE OF (SIR_event_assignment);
   evaluated_items: SET[1:?] OF TAS_orbit_arc;
WHERE
   wr1: SELF\SIR_event_assignment.role = 'evaluation';
END_ENTITY;
(*
```

Attribute definitions:

evaluated_items: set of one or more **TAS_orbit_arc** that shall be evaluated at the **SIR_space_mission_aspect** referred to as assigned_event;

Formal propositions:

wr1: the name of the role of the **event_assignment** shall be **'evaluation'**.

5.2.3.24 TAS_orbit_governor

A **TAS_orbit_governor** is a relationship between a **property_definition** that characterizes the shape of an orbit arc and the **property_definition** characterizing a celestial body. The relationship states that the celestial body governs the orbit arc.

EXPRESS specification:

```
*)
ENTITY TAS_orbit_governor
SUBTYPE OF (property_definition_relationship);
WHERE
   WR1: description = 'governing celestial body';
   WR2: relating_property_definition.description = 'gravitational data';
   WR3: SIZEOF(QUERY(prop_rep <*
USEDIN(SELF\property_definition_relationship.related_property_definition,'TAS_SCHE
MA.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | NOT
('TAS_SCHEMA.TAS_ORBIT_ARC' IN TYPEOF(prop_rep)))) =0;
   WR4: 'TAS_SCHEMA.TAS_CELESTIAL_BODY' IN
relating_property_definition.definition.formation.of_product;
END_ENTITY;
(*
```

Formal propositions:

WR1: description shall be the string 'governing celestial body';

WR2: relating_property_definition.description shall be the string 'gravitational data';

WR3: related_property_definition shall be a **property_definition** associated with a **TAS_orbit_arc**.

WR4: relating_property_definition shall refer to a **property_definition** assigned to a **TAS_celestial_body**.

5.2.3.25 TAS_orbit_assignment

A **TAS_orbit_assignment** is a relationship between a **property_definition** that characterizes the shape of an orbit arc and a **property_definition** characterizing a **TAS_celestial_body_with_orbit**. The relationship states that the **TAS_celestial_body_with_orbit** moves on the orbit arc.

EXPRESS specification:

```
*)
ENTITY TAS_orbit_assignment
SUBTYPE OF (property_definition_relationship);
WHERE
   WR1: description = 'orbit assignment';
   WR2: related_property_definition.description = 'gravitational data';
   WR3: SIZEOF(QUERY(prop_rep <*
USEDIN(SELF\property_definition_relationship.relating_property_definition,'TAS_SCH
EMA.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | NOT
('TAS_SCHEMA.TAS_ORBIT_ARC' IN TYPEOF(prop_rep)))) =0;
   WR4: 'TAS_SCHEMA.TAS_CELESTIAL_BODY_WITH_ORBIT' IN
related_property_definition.definition.formation.of_product;
END_ENTITY;
(*
```

Formal propositions:

WR1: description shall be the string 'orbit assignment';

WR2: related_property_definition.description shall be the string 'gravitational data';

WR3: relating_property_definition shall be a **property_definition** assocuated with a **TAS_orbit_arc**.

WR4: related_property_definition shall refer to a **property_definition** assigned to a **TAS_celestial_body_with_orbit**.

5.2.3.26 TAS_orbit_arc

A **TAS_orbit_arc** specifies the generic information of an orbit arc. It is an abstract supertype of one of **TAS_discretised_orbit_arc** and **TAS_kepler_orbit_arc**.

A **TAS_orbit_arc** shall be referred to by at least one **SIR_mission_aspect**, in order to get its context of validity.

EXPRESS specification:

```
*)
ENTITY TAS_orbit_arc
ABSTRACT SUPERTYPE OF (ONEOF(TAS_discretised_orbit_arc, TAS_kepler_orbit_arc))
SUBTYPE OF (property_definition_representation);
DERIVE
   geom_rep: shape_representation := SELF.used_representation;
   orbit: characterized_object := SELF.definition.definition;
INVERSE
   evaluations: SET[0:?] OF TAS_evaluation_event FOR evaluated_items;
   space_mission_aspect: SET[1:?] OF SIR_space_mission_aspect FOR
characterized_objects;
WHERE
   WR1: orbit\characterized_object.description IN [‘general’,’geostationary’,’sun
synchronous’,’molniya’];
   WR2: SIZEOF(QUERY( pf <*
USEDIN(SELF,’TAS_SCHEMA.PROPERTY_DEFINITION.DEFINITION’) |
SIZEOF(QUERY(pfr <*
USEDIN(pf,’TAS_SCHEMA.PROPERTY_DEFINITION_RELATIONSHIP.RELATED_PROPERTY_DEFINITION
’) | ‘TAS_SCHEMA.TAS_ORBIT_GOVERNOR’ IN TYPEOF(pfr) )) >0 )) =1;
   WR3: ‘TAS_SCHEMA.TAS_SPACE_COORDINATE_SYSTEM’ IN
TYPEOF(geom_rep.context_of_items);
   WR4: ((geom_rep\representation.context_of_items.context_type IN [‘planet
centric star fixed ecliptic’, ‘planet centric star fixed equatorial’]) AND
 (SIZEOF(QUERY(it <* geom_rep\represntation.items | (it.name = ‘star spherical
right ascension’) AND
 (‘TAS_SCHEMA.PLANE_ANGLE_MEASURE_WITH_UNIT’ IN TYPEOF(it)) )) =1)) XOR
(SIZEOF(QUERY(it <* geom_rep\represntation.items | (it.name = ‘star spherical right
ascension’) AND (‘TAS_SCHEMA.PLANE_ANGLE_MEASURE_WITH_UNIT’ IN TYPEOF(it)) )) =0);
   WR5: SIZEOF(QUERY(it <* geom_rep\represntation.items | (it.name = ‘star
spherical declination’) AND (‘TAS_SCHEMA.PLANE_ANGLE_MEASURE_WITH_UNIT’ IN
TYPEOF(it)) )) <=1;
   WR6: SIZEOF(QUERY(it <* geom_rep\represntation.items |
(‘TAS_SCHEMA.TAS_REVOLUTION_PERIOD’ IN TYPEOF(it)) )) <=1;
END_ENTITY;
(*
```

Attribute definitions:

geom_rep: derived attribute identifying the **shape_representation** which contains the data defining the shape of the orbit arc;

orbit: derived attribute that identifies the **characterized_object** that is depicted by the **TAS_orbit_arc**;

evaluations: specifies a set of 0 to many **TAS_evaluation_event** that each identifies a space mission event where evaluation shall take place;

space_mission_aspect: specifies a set of one or more **SIR_mission_aspect**s that refer to the **TAS_orbit_arc** through its attribute characterized_objects.

Formal propositions:

WR1: the description attribute of orbit shall have one of the following values: ‘general’, ’geostationary’, ’sun synchronous’ or ’molniya’;

WR2: the **TAS_orbit_arc** shall be referred to by extaly one **TAS_orbit_governor** as related_property_definition.definition;

WR3: a **TAS_orbit_arc** shall have a **TAS_space_coordinate_system** as context of geom_rep.

WR4: if the coordinate system is ‘planet centric fixed ecliptic’ or ‘planet centric fixed equatorial’ then geom_rep shall contain one item with name ‘star spherical right ascension’ and of type **plane_angle_measure_with_unit**. Otherwise geom_rep shall not contain such an item;

WR5: geom_rep shall contain at most one item with name ‘star spherical declination’ and of type **plane_angle_measure_with_unit**;

WR6: geom_rep shall contain at most one item of type **TAS_revolution_period**.

5.2.3.27 TAS_discretised_orbit_arc

A **TAS_discretised_orbit_arc** entity is a specialization of **TAS_orbit_arc** which defines the orbit as a list of computation positions and velocity vectors. A set of Kepler orbit parameters may be associated for information, in a representation associated with the representation defining the orbit.

EXPRESS specification:

```
*)
ENTITY TAS_discretised_orbit_arc
SUBTYPE OF (TAS_orbit_arc);
WHERE
   wr1: SIZEOF(QUERY(repr_item <*
SELF\TAS_orbit_arc.geom_rep\representation.items |
SIZEOF(['TAS_SCHEMA.TAS_POSITION_AND_VELOCITY',
'TAS_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM']*TYPEOF(repr_item))<>2)) =0;
   wr2: TAS_valid_associated_kepler_set(SELF);
END_ENTITY;
(*
```

Formal propositions:

wr1: the items of **geom_rep** shall contain a **TAS_position_and_velocity** and a **descriptive_representation.item**;

wr2: in the case where the **representation** is referred to by a **representation_relationship**, then the attribute **representation_relationship.description** shall be equal to ‘association with unperturbed_orbit_data’ and the items of the related representation shall be a **TAS_kepler_set** instance.

5.2.3.28 TAS_kepler_orbit_arc

A **TAS_kepler_orbit_arc** is a specialisation of **TAS_orbit_arc** where the orbit arc is defined by a set of keplerian parameters (**TAS_kepler_set**).

EXPRESS specification:

```
*)
ENTITY TAS_kepler_orbit_arc
SUBTYPE OF (TAS_orbit_arc);
WHERE
   wr1: SIZEOF(SELF\TAS_orbit_arc.geom_rep\representation.items) = 1;
   wr2: SIZEOF(QUERY(rep_item <* SELF\TAS_orbit_arc.geom_rep\representation.items
| NOT ('TAS_SCHEMA.TAS_KEPLER_SET' IN TYPEOF(rep_item)))) =0;
END_ENTITY;
(*
```

Formal propositions:

wr1: the **representation** associated with the orbit shall have exactly one item;

wr2: this item shall be a **TAS_kepler_set**.

5.2.3.29 TAS_celestial_body

A **TAS_celestial_body** is a celestial body, as depicted within the context of thermal radiative analyses.

EXPRESS specification:

```
*)
ENTITY TAS_celestial_body
SUBTYPE OF (product);
INVERSE
   category: product_related_product_category FOR products;
WHERE
   wr1: category\product_category.name = 'celestial bodies';
   wr2: SIZEOF(QUERY(pc <* SELF.frame_of_reference | (pc.discipline_type =
'orbitography') AND (pc\application_context_element.name = 'thermal radiative
studies') AND (pc\application_context_element.frame_of_reference.application =
'space products development'))) =1;
   WR3: SIZEOF(QUERY(pdf <* USEDIN(SELF,
'TAS_SCHEMA.PRODUCT_DEFINITION_FORMATION.OF_PRODUCT')|
'TAS_SCHEMA.SIR_PRODUCT_VERSION' IN TYPEOF(pdf)))>0;
   WR4: valid_properties_of_celestial_body(SELF);
END_ENTITY;
(*
```

Attribute definition:

category: specifies the **product_related_product_category** that is the category which a **TAS_celestial_body** shall belong to;

Formal propositions:

WR1: the name of the category shall be 'celestial bodies';

WR2: there shall be one element in frame_of_reference such:

- the referred application context is 'space products development';
- the referred **product_context** has 'thermal radiative analyses' for name and 'orbitography' for discipline_type;

WR3: there shall be at least one **SIR_product_version** that refers to the **TAS_celestial_body**;

WR4: a **TAS_celestial_body** shall have one set of gravitational characteristics (i.e. it shall be related to one **TAS_gravitational_aspects_of_celestial_body**) and at most one set of thermal characteristics (i.e. it shall be related to at most one **TAS_thermal_aspects_of_celestial_body**), per instance of **SIR_product_version** referring to it.

5.2.3.30 TAS_celestial_body_with_orbit

A **TAS_celestial_body_with_orbit** is a kind of **TAS_celestial_body** for which an orbit is specified as a **TAS_orbit_arc**.

EXPRESS specification:

```
*)
ENTITY TAS_celestial_body_with_orbit
SUBTYPE OF (TAS_celestial_body);
WHERE
   WR1: orbit_assigned_to(SELF);
END_ENTITY;
(*
```

Formal proposition:

WR1: specifies that each **TAS_celestial_body_with_orbit** is associated with one **TAS_orbit_arc** through an instance of **TAS_orbit_assignment**.

5.2.3.31 TAS_gravitational_aspects_of_celestial_body

A **TAS_gravitational_aspects_of_celestial_body** is the identification of the gravitational data characterizing a celestial body.

EXPRESS specification:

```
*)
ENTITY TAS_gravitational_aspects_of_celestial_body
SUBTYPE OF (property_definition);
INVERSE
   prop_rep: property_definition_representation FOR definition;
WHERE
   WR1:
SELF\property_definition.definition.frame_of_reference\application_context_element
.name = 'gravitation related studies';
   WR2: SELF\property_definition.description = 'gravitational characteristics';
   WR3: SIZEOF(QUERY(rep_it <* prop_rep.used_representation.items |
('TAS_SCHEMA.TAS_RADIUS_OF_CELESTIAL_BODY' IN TYPEOF(rep_it)))) =1;
   WR4: SIZEOF(QUERY(rep_it <* prop_rep.used_representation.items |
('TAS_SCHEMA.TAS_GRAVITATIONAL_CONSTANT' IN TYPEOF(rep_it)))) =1;
   WR5: SIZEOF(QUERY(rep_it <* prop_rep.used_representation.items |
('TAS_SCHEMA.TAS_REVOLUTION_PERIOD' IN TYPEOF(rep_it)))) <=1;
   WR6: prop_rep.used_representation.context_of_items.context_type = 'numerical
data';
   WR7: 'TAS_SCHEMA.TAS_CELESTIAL_BODY' IN
TYPEOF(SELF\property_definition.definition.formation.of_product);
END_ENTITY;
(*
```

Attribute definitions:

prop_rep: specifies the **property_definition_representation** that shall refer to the entity through its attribute definition.

Formal propositions:

WR1: the name of the context of the **product_definition** referred to through the attribute definition, shall be 'gravitation related studies';

WR2: the description attribute of the entity shall be 'gravitational characteristics';

WR3: the representation associated with prop_rep shall contain one **TAS_radius_of_celestial_body**;

WR4: the representation associated with prop_rep shall contain one **TAS_gravitational_constant**;

WR5: the representation associated with prop_rep shall contain at most one **TAS_sideral_rotation_period**;

WR6: the context type of the representation associated with the **TAS_gravitational_aspects_of_celestial_body** shall be 'numerical data';

WR7: the **TAS_gravitational_aspects_of_celestial_body** shall be assigned to a **TAS_celestial_body**.

5.2.3.32 TAS_thermal_aspects_of_celestial_body

A **TAS_thermal_aspects_of_celestial_body** is the identification of the thermal data characterizing a celestial body.

EXPRESS specification:

```
*)
ENTITY TAS_thermal_aspects_of_celestial_body
SUBTYPE OF (property_definition);
INVERSE
   prop_rep: property_definition_representation FOR definition;
WHERE
   WR1:
SELF\property_definition.definition.frame_of_reference\application_context_element
.name = 'thermal studies';
   WR2: SELF\property_definition.description = 'thermal characteristics';
   WR3: SIZEOF(QUERY(rep_it <* prop_rep.used_representation.items |
('TAS_SCHEMA.TAS_BLACK_BODY_SURFACE_TEMPERATURE' IN TYPEOF(rep_it)))) =1;
   WR4: SIZEOF(QUERY(rep_it <* prop_rep.used_representation.items |
('TAS_SCHEMA.TAS_NOTIONAL_DISTANCE' IN TYPEOF(rep_it)))) <=1;
   WR5: SIZEOF(QUERY(rep_it <* prop_rep.used_representation.items |
('TAS_SCHEMA.TAS_THERMAL_RADIATION_SPECTRUM' IN TYPEOF(rep_it)))) <=1;
```

```
   WR6: prop_rep.used_representation.context_of_items.context_type = 'numerical
data';
   WR7: 'TAS_SCHEMA.TAS_CELESTIAL_BODY' IN
TYPEOF(SELF\property_definition.definition.formation.of_product);
END_ENTITY;
(*
```

Attribute definitions:

prop_rep: specifies the **property_definition_representation** that shall refer to the entity through the attribute definition.

Formal propositions:

WR1: the name of the context of the **product_definition** referred to through the attribute definition, shall be 'thermal studies';

WR2: the description attribute of the entity shall be 'thermal characteristics';

WR3: the representation associated with prop_rep shall contain one **TAS_black_body_surface_temperature**;

WR4: the representation associated with prop_rep shall contain at most one **TAS_notional_distance**;

WR5: the representation associated with prop_rep shall contain at most one **TAS_thermal_radiation_spectrum**;

WR6: the context type of the representation associated with the **TAS_thermal_aspects_of_celestial_body** shall be 'numerical data';

WR7: the **TAS_thermal_aspects_of_celestial_body** shall be assigned to a **TAS_celestial_body**.

5.2.3.33 TAS_albedo_reflector

A **TAS_albedo_reflector** entity specifies the parameters that define thermal radiation through albedo reflection.

It is a subtype of **product_definition_relationship**, the **relating_product_definition** being the **product_definition** defining the thermal aspect of the considered celestial body and the **related_product_definition** being a thermal aspect of another celestial body (from which the albedo reflection originates).

This entity has one mandatory **property_definition** (albedo_reflection_factor) and an optional one (thermal radiation spectrum).

EXPRESS specification:

```
*)
ENTITY TAS_albedo_reflector
SUBTYPE OF (product_definition_relationship);
INVERSE
   properties: SET[1:?] OF property_definition FOR definition;
WHERE
   wr1: TAS_valid_albedo_reflector(SELF);
END_ENTITY;
(*
```

Attribute definitions:

properties: inverse attribute defining the set of instances of **property_definition** assigned to the **TAS_albedo_reflector**.

Formal propositions:

wr1: the considered instance shall have the required structure and properties.

5.2.3.34 TAS_albedo_reflection_data

A **TAS_albedo_reflection_data** is the association of a **representation** containing the data relevant for the representation of the characteristics of a **TAS_albedo_reflector** with the **property_definition** that refers to a **TAS_albedo_reflector**.

EXPRESS specification:

```
*)
ENTITY TAS_albedo_reflection_data
SUBTYPE OF (property_definition_representation);
WHERE
   wr1: SELF\property_definition_representation.definition.description = 'albedo
reflection characteristics'
   wr2: 'TAS_SCHEMA.TAS_ALBEDO_REFLECTOR' IN
TYPEOF(SELF\property_definition_representation.definition.definition);
   wr3: SIZEOF(QUERY(it<*
SELF\property_definition_representation.used_representation.items |
'TAS_SCHEMA.TAS_ALBEDO_REFLECTION_FACTOR' IN TYPEOF(it))) =1;
   wr4: SIZEOF(QUERY(it<*
SELF\property_definition_representation.used_representation.items |
'TAS_SCHEMA.TAS_THERMAL_RADIATION_SPECTRUM' IN TYPEOF(it))) <=1;
END_ENTITY;
(*
```

Formal propositions:

wr1: the description attribute of the **property_definition** referred to as definition, shall be 'albedo reflection characteristics';

wr2: the entity referred to by this **property_definition** shall be a **TAS_albedo_reflector**;

wr3: the **representation** referred to as used_representation, shall contain one **TAS_albedo_reflection_factor**;

wr4: the **representation** referred to as used_representation, shall contain at most one **TAS_thermal_radiation_spectrum**;

5.2.3.35 TAS_body_motion_representation

A **TAS_body_motion_representation** is a kind of **property_definition_representation** that associates a **representation** that depicts a rotation motion with a **property_definition** asigned to a space product.

A **TAS_body_motion_representation** shall be referred to by at least one **SIR_space_mission_aspect**, in order to get its context of validity.

EXPRESS specification:

```
*)
ENTITY TAS_body_motion_representation
SUBTYPE OF (property_definition_representation);
INVERSE
   space_mission_aspects: SET[1:?] OF SIR_space_mission_aspect FOR
characterized_objects;
WHERE
   wr1: SIZEOF(QUERY(it<*
SELF\property_definition_representation.used_representation.items |
('TAS_SCHEMA.SIR_ROTATION_MOTION') IN TYPEOF(it)))=1;
   wr2: SELF\property_definition_representation.definition.description =
'continuous motion';
   wr3:
SIZEOF(['TAS_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT','TAS_SCHEMA.GLOBAL_UNIT_ASSI
GNED_CONTEXT']*TYPEOF(SELF\property_definition_representation.used_representation.
context_of_items))=2;
END_ENTITY;
(*
```

Attribute definitions:

space_mission_aspects: specifies a set of one or more **SIR_space_mission_aspect** that refer to the **TAS_body_motion_representation**.

Formal propositions:

wr1: the **representation** referred to as used_representation shall contain one **SIR_rotation_motion** in its items;

wr2: the **property_definition** referred to as definition shall have ‘continous motion’ for name.

wr3: the context of the **representation**,referred to as used_representation, shall be a **geometric_representation_context** and a **global_unit_assigned_context**.

5.2.3.36 TAS_thermal_environment

A **TAS_thermal_environment** entity specifies is a kind of **data_environment** that defines a thermal environment.

It shall be referred to by at least one **SIR_space_mission_aspect** through its attribute characterized_objects.

EXPRESS specification:

```
*)
ENTITY TAS_thermal_environment
SUBTYPE OF (data_environment);
INVERSE
   space_mission_aspects: SET[1:?] OF SIR_space_mission_aspect FOR
characterized_objects;
WHERE
   wr1: SIZEOF(QUERY(pdr <* SELF\data_environment.elements |
‘TAS_SCHEMA.TAS_THERMAL_ENVIRONMENT_DATA’ IN TYPEOF(pdr))) =1;
END_ENTITY;
(*
```

Attribute definitions:

space_mission_aspects: specifies a set of one or more **SIR_space_mission_aspect** that refer to the **TAS_thermal_environment**.

Formal propositions:

wr1: there shall be one instance of **TAS_thermal_environment_data** in the set of elements.

5.2.3.37 TAS_thermal_environment_data

A **TAS_thermal_environment_data** is the association of a **representation** containing the data relevant for the representation of the characteristics of a thermal environment with the **property_definition** referring to the characterized_object that identifies the thermal_environment.

EXPRESS specification:

```
*)
ENTITY TAS_thermal_environment_data
SUBTYPE OF (property_definition_representation);
WHERE
   wr1: SELF\property_definition_representation.definition.description = ‘thermal
environment characteristics’;
   wr2: ‘TAS_SCHEMA.CHARACTERIZED_OBJECT’ IN
TYPEOF(SELF\property_definition_representation.definition.definition);
   wr3: SIZEOF(QUERY(it<*
SELF\property_definition_representation.used_representation.items |
‘TAS_SCHEMA.THERMODYNAMIC_TEMPERATURE_MEASURE_WITH_UNIT’ IN TYPEOF(it))) =1;
   wr4: SIZEOF(QUERY(it<*
SELF\property_definition_representation.used_representation.items |
‘TAS_SCHEMA.TAS_THERMAL_FLUX’ IN TYPEOF(it))) <=1;
```

```
END_ENTITY;
(*
```

Formal propositions:

wr1: the description attribute of the **property_definition** referred to as definition, shall be 'thermal environment characteristics';

wr2: the entity referred to by this **property_definition** shall be a **characterized_object**;

wr3: the **representation** referred to as used_representation, shall contain one **thermodynamic_temperature_measure_with_unit**;

wr4: the **representation** referred to as used_representation, shall contain at most one **TAS_thermal_flux**;

5.2.3.38 TAS_material_property_environment

A **TAS_material_property_environment** entity specifies is a kind of **data_environment** that defines a consistent set of conditions relevant for studies where the material characteristics of a space product are considered.

It shall be referred to by at least one **SIR_space_mission_aspect** through its attribute characterized_objects.

EXPRESS specification:

```
*)
ENTITY TAS_material_property_environment
SUBTYPE OF (data_environment);
INVERSE
   space_mission_aspects: SET[1:?] OF SIR_space_mission_aspect FOR
characterized_objects;
END_ENTITY;
(*
```

Attribute definitions:

space_mission_aspects: specifies a set of one or more **SIR_space_mission_aspect** that refer to the **TAS_material_environment**.

5.2.3.39 TAS_basic_thermal_radiative_model

A **TAS_basic_thermal_radiative_model** is a kind of **SIR_model** that only contains **SIR_thermal_radiative_face_set**s.

EXPRESS specification:

```
*)
ENTITY TAS_basic_thermal_radiative_model
SUBTYPE OF (SIR_model);
   SELF\representation.items: SET[1:?] OF SIR_thermal_radiative_face_set;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation.items: specifies a set of one or more **SIR_thermal_radiative_face_set** that are the elements of the **TAS_basic_thermal_radiative_model**.

5.2.3.40 TAS_meshed_thermal_radiative_model

A **TAS_meshed_thermal_radiative_model** is a kind of **SIR_model** that only contains **SIR_thermal_radiative_meshing**s.

EXPRESS specification:

```
*)
```

```
ENTITY TAS_meshed_thermal_radiative_model
SUBTYPE OF (SIR_model);
   SELF\representation.items: SET[1:?] OF SIR_thermal_radiative_meshing;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation.items: specifies a set of one or more **SIR_thermal_radiative_meshing** that are the elements of the **TAS_meshed_thermal_radiative_model**.

5.2.3.41 TAS_enclosure

A **TAS_enclosure** is a group of **TAS_thermal_radiative_face_set** that cannot exchange heat through thermal radiation with **TAS_thermal_radiative_face_set** belonging to another **TAS_enclosure**.

EXPRESS specification:

```
*)
ENTITY TAS_enclosure
SUBTYPE OF (group_assignment);
   grouped_items: SET[1:?] OF SIR_thermal_radiative_face_set;
END_ENTITY;
(*
```

Attribute definitions:

grouped_items: set of one or more **SIR_thermal_radiative_face_set** belonging to the enclosure.

5.2.3.42 TAS_material_property

The **TAS_material_property** entity specifies a physical property associated with a given material. This property is defined in a given **TAS_material_property_environment**.

A **TAS_material_property** is related to the **material_designation** that identifies the material of which the **TAS_material_property** carries a portion of the physical characterization.

EXPRESS specification:

```
*)
ENTITY TAS_material_property
SUBTYPE OF (material_property_representation);
WHERE
   wr1: SIZEOF(QUERY(it <*
SELF\property_definition_representation.used_representation.items |
NOT('TAS_SCHEMA.TAS_PROPERTY_VALUE' IN TYPEOF(it)) )) =0;
   wr2: 'TAS_SCHEMA.TAS_MATERIAL_PROPERTY_ENVIRONMENT' IN
TYPEOF(SELF\material_property_representation.dependent_environment);
   wr3:
SIZEOF(USEDIN(SELF,'TAS_SCHEMA.MATERIAL_DESIGNATION_CHARACTERIZATION.PROPERTY'))
=1;
END_ENTITY;
(*
```

Formal propositions:

wr1: the items of the representation referred to by the entity as used_representation shall all be **TAS_property_value**s;

wr2: the dependent_environment shall be a **TAS_material_property_environment**.

wr3: the **TAS_material_property** shall be referred to by exactly one **material_designation_characterization**.

5.2.4 TAS_schema imported entity modifications

5.2.4.1 action

The base definition of the **action** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **action** entity:

- subtype_mandatory_action;

5.2.4.2 action_relationship

The base definition of the **action_relationship** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **action_relationship** entity:

- subtype_mandatory_action_relationship;

5.2.4.3 action_assignment

The base definition of the **action_assignment** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **action_assignment** entity:

- exclusive_subtypes_action_assignment;

5.2.4.4 application_context

The base definition of the **application_context** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **application_context** entity:

- application_context_requires_ap_definition

5.2.4.5 application_protocol_definition

The base definition of the **application_protocol_definition** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **application_protocol_definition** entity:

- application_context_requires_ap_definition

5.2.4.6 approval

The base definition of the **approval** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **approval** entity:

- approval_requires_approval_date_time;
- approval_requires_approval_person_organization;
- approval_are_assigned;

5.2.4.7 approval_date_time

The base definition of the **approval_date_time** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **approval_date_time** entity:

- approval_requires_approval_date_time;
- approval_date_time_requires_date_time;

5.2.4.8 approval_person_organization

The base definition of the **approval_person_organization** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **approval_person_organization** entity:

- approval_requires_approval_person_organization;
- approval_person_organization_requires_person_organization;

5.2.4.9 approval_role

The base definition of the **approval_role** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **approval_role** entity:

- dependent_instantiable_approval_role;

5.2.4.10 approval_status

The base definition of the **approval_status** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **approval_status** entity:

- dependent_instantiable_status;

5.2.4.11 date

The base definition of the **date** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **date** entity:

- dependent_instantiable_date;
- subtype_mandatory_date;

5.2.4.12 date_time_role

The base definition of the **date_time_role** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **date_time_role** entity:

- dependent_instantiable_date_time_role;

5.2.4.13 person

The base definition of the **person** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **person** entity:

- dependent_instantiable_person;

5.2.4.14 person_and_organization_role

The base definition of the **person_and_organization_role** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **person_and_organization_role** entity:

- dependent_instantiable_person_and_organization_role;

5.2.4.15 security_classification_level

The base definition of the **security_classification_level** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **security_classification_level** entity:

- dependent_instantiable_security_classification_level;

5.2.4.16 type_qualifier

The base definition of the **type_qualifier** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **type_qualifier** entity:

- dependent_instantiable_type_qualifier;

5.2.4.17 data_environment

The base definition of the **data_environment** entity is given in ISO 10303-45. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **data_environment** entity:

- subtype_mandatory_data_environment;

5.2.4.18 product_context

The base definition of the **product_context** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **product_context** entity:

- product_context_requires_product;

5.2.4.19 product_definition_context

The base definition of the **product_definition_context** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **product_definition_context** entity:

- prod_def_context_requires_prod_def;

5.2.4.20 product

The base definition of the **product** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **product** entity:

- product_requires_version;
- product_context_requires_product;

5.2.4.21 SIR_product_version

The base definition of the **SIR_product_version** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_product_version** entity:

- product_requires_version;

5.2.4.22 product_definition_formation

The base definition of the **product_definition_formation** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **product_definition_formation** entity:

- subtype_mandatory_product_definition_formation;

5.2.4.23 product_definition_relationship

The base definition of the **product_definition_relationship** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **product_definition_relationship** entity:

- subtype_mandatory_product_definition_relationship;

5.2.4.24 product_definition

The base definition of the **product_definition** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **product_definition** entity:

- prod_def_context_requires_prod_def;

5.2.4.25 global_unit_assigned_context

The base definition of the **global_unit_assigned_context** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **global_unit_assigned_context** entity:

- TAS_global_units

5.2.4.26 geometric_representation_context

The base definition of the **geometric_representation_context** entity is given in ISO 10303-42. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **geometric_representation_context** entity:

- TAS_dimensionality_is_3

5.2.4.27 shape_representation

The base definition of the **shape_representation** entity is given in ISO 10303-41. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **shape_representation** entity:

- TAS_point_id_unique

5.2.4.28 cartesian_point

The base definition of the **cartesian_point** entity is given in ISO 10303-42. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **cartesian_point** entity:

- TAS_point_id_unique

5.2.4.29 advanced_face

The base definition of the **advanced_face** entity is given in ISO 10303-511. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **advanced_face** entity:

- at_most_2_faces_per_surface;

5.2.4.30 SIR_thermal_radiative_face_set

The base definition of the **SIR_thermal_radiative_face_set** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_thermal_radiative_face_set** entity:

- at_most_2_faces_per_surface;
- thermal_face_requires_thermal_model;

5.2.4.31 SIR_ato_campaign

The base definition of the **SIR_ato_campaign** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_ato_campaign** entity:

- ato_campaign_requires_creation_date;
- ato_campaign_requires_contact;
- at_most_one_modification_date;
- at_most_one_security_level;

5.2.4.32 SIR_model

The base definition of the **SIR_model** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_model** entity:

- at_most_one_security_level;
- model_requires_definition;
- unique_ids_in_model;

5.2.4.33 SIR_node

The base definition of the **SIR_node** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_node** entity:

- at_most_one_security_level;
- unique_ids_in_model;

5.2.4.34 SIR_node_relationship

The base definition of the **SIR_node_relationship** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_node_relationship** entity:

- at_most_one_security_level;
- unique_ids_in_model;

5.2.4.35 SIR_submodel_usage

The base definition of the **SIR_submodel_usage** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_submodel_usage** entity:

- unique_ids_in_model;

5.2.4.36 SIR_aspect

The base definition of the **SIR_aspect** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_aspect** entity:

- at_most_one_security_level;

5.2.4.37 SIR_ato_case

The base definition of the **SIR_ato_case** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_ato_case** entity:

- at_most_one_security_level;

5.2.4.38 SIR_ato_phase

The base definition of the **SIR_ato_phase** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_ato_phase** entity:

- at_most_one_security_level;

5.2.4.39 SIR_component_sequence

The base definition of the **SIR_component_sequence** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_component_sequence** entity:

- component_sequence_requires_aspect;

5.2.4.40 SIR_boolean_result_surface

The base definition of the **SIR_boolean_result_surface** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_boolean_result_surface** entity:

- restrict_boolean_operator;

5.2.4.41 SIR_space_mission_aspect

The base definition of the **SIR_space_mission_aspect** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_space_mission_aspect** entity:

- restrict_mission_aspects;

5.2.4.42 SIR_kinematic_link

The base definition of the **SIR_kinematic_link** entity is given in SIR. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **SIR_kinematic_link** entity:

- kinematic_link_requires_shape;
- restrict_represented_product;

5.2.4.43 kinematic_link

The base definition of the **kinematic_link** entity is given in ISO 10303-105. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **kinematic_link** entity:

- subtype_mandatory_kinematic_link;

5.2.4.44 mechanism

The base definition of the **mechanism** entity is given in ISO 10303-105. The following modifications apply to this protocol.

Associated global rules:

The following global rules apply to the **mechanism** entity:

- mechanism_requires_ground_representation;

5.2.5 TAS_schema rule definitions

5.2.5.1 ato_campaign_requires_creation_date

The **ato_campaign_requires_creation_date** specifies that each instance of **SIR_ato_campaign** shall be referred to by one instance of **TAS_date_assignment** that relates it to its creation date.

EXPRESS specification:

```
*)
RULE ato_campaign_requires_creation_date FOR (SIR_ato_campaign,
TAS_date_assignment);
WHERE
   WR1: SIZEOF(QUERY(obj<* SIR_ato_campaign| SIZEOF(QUERY(nda <*
TAS_date_assignment| (obj IN nda.items) AND (nda.role.name = 'creation_date')))<>1
))= 0;
END_RULE;
(*
```

Argument definitions:

SIR_ato_campaign: specifies the set of all instances of **SIR_ato_campaign**;

TAS_date_assignment: specifies the set of all instances of **TAS_date_assignment**.

Formal proposition:

WR1: there shall not be any instance of **SIR_ato_campaign** that is not referred to by exactly one **TAS_date_assignment** having the role 'CREATION_DATE'.

5.2.5.2 ato_campaign_requires_contact

The **ato_campaign_requires_creation_date** specifies that each instance of **SIR_ato_campaign** shall be referred to by one instance of **TAS_contact_assignment** that relates it to a contact person/organization.

EXPRESS specification:

```
*)
RULE ato_campaign_requires_contact FOR (SIR_ato_campaign, TAS_contact_assignment);
WHERE
   WR1: SIZEOF(QUERY(obj<* SIR_ato_campaign| SIZEOF(QUERY(nca <*
TAS_contact_assignment | obj IN nca.items))<>1 ))= 0;
END_RULE;
```

```
(*
```

Argument definitions:

SIR_ato_campaign: specifies the set of all instances of **SIR_ato_campaign**;

TAS_contact_assignment: specifies the set of all instances of **TAS_contact_assignment**;

Formal proposition:

WR1: there shall not be any instance of **SIR_ato_campaign** that is not referred to by exactly one **TAS_contact_assignment**.

5.2.5.3 at_most_one_modification_date

The **ato_campaign_requires_creation_date** specifies that each instance of **SIR_ato_campaign** shall be referred to by at most one instance of **SIR_ato_campaign** that relates it to a modification date.

EXPRESS specification:

```
*)
RULE at_most_one_modification_date FOR (SIR_ato_campaign, TAS_date_assignment);
WHERE
   WR1: SIZEOF(QUERY(obj<* SIR_ato_campaign| SIZEOF(QUERY(nda <*
TAS_date_assignment| (obj IN nda.items) AND (nda.role.name =
'last_modification_date')))>1 ))= 0;
END_RULE;
(*
```

Argument definitions:

SIR_ato_campaign: specifies the set of all instances of **SIR_ato_campaign**;

TAS_date_assignment: specifies the set of all instances of **TAS_date_assignment**.

Formal proposition:

WR1: there shall not be any instance of **SIR_ato_campaign** that is not referred to more than one **TAS_date_assignment** having the role 'LAST_MODIFICATION_DATE'.

5.2.5.4 at_most_one_security_level

The rule **at_most_one_security_level** specifies that each instance of **SIR_model**, **SIR_node**, **SIR_node_relationship**, **SIR_ato_campaign**, **SIR_ato_case** and **SIR_ato_phase** shall be referred to by at most one instance of **TAS_security_assignment** that relates it to a security level.

EXPRESS specification:

```
*)
RULE at_most_one_security_level FOR (SIR_model, SIR_node, SIR_node_relationship,
SIR_aspect, SIR_ato_campaign, SIR_ato_case, SIR_ato_phase,
TAS_security_assignment);
WHERE
   WR1: SIZEOF(QUERY(obj<* SIR_model| SIZEOF(QUERY(nsa <* TAS_security_assignment
| obj IN nsa.items))>1 ))= 0;
   WR2: SIZEOF(QUERY(obj<* SIR_node| SIZEOF(QUERY(nsa <* TAS_security_assignment |
obj IN nsa.items))>1 ))= 0;
   WR3: SIZEOF(QUERY(obj<* SIR_node_relationship| SIZEOF(QUERY(nsa <*
TAS_security_assignment | obj IN nsa.items))>1 ))= 0;
   WR4: SIZEOF(QUERY(obj<* SIR_ato_campaign| SIZEOF(QUERY(nsa <*
TAS_security_assignment | obj IN nsa.items))>1 ))= 0;
   WR5: SIZEOF(QUERY(obj<* SIR_ato_case| SIZEOF(QUERY(nsa <*
TAS_security_assignment | obj IN nsa.items))>1 ))= 0;
   WR6: SIZEOF(QUERY(obj<* SIR_ato_phase| SIZEOF(QUERY(nsa <*
TAS_security_assignment | obj IN nsa.items))>1 ))= 0;
END_RULE;
(*
```

Argument definitions:

SIR_model: specifies the set of all instances of **SIR_model**.

SIR_node: specifies the set of all instances of **SIR_node**.

SIR_node_relationship: specifies the set of all instances of **SIR_node_relationship**.

SIR_ato_campaign: specifies the set of all instances of **SIR_ato_campaign**.

SIR_ato_case: specifies the set of all instances of **SIR_ato_case**.

SIR_ato_phase: specifies the set of all instances of **SIR_ato_phase**.

TAS_date_assignment: specifies the set of all instances of **TAS_date_assignment**.

Formal proposition:

WR1: there shall not be more than one security level assigned to a **SIR_model**;

WR2: there shall not be more than one security level assigned to a **SIR_node**;

WR3: there shall not be more than one security level assigned to a **SIR_node_relationship**;

WR4: there shall not be more than one security level assigned to a **SIR_ato_campaign**;

WR5: there shall not be more than one security level assigned to a **SIR_ato_case**;

WR6: there shall not be more than one security level assigned to a **SIR_ato_phase**;

5.2.5.5 application_context_requires_ap_definition

The rule **application_context_requires_ap_definition** specifies that any instance of **application_context** shall be referred to by exactly one instance of **application_protocol_definition**.

EXPRESS specification:

```
*)
RULE application_context_requires_ap_definition FOR
  (application_context, application_protocol_definition);
WHERE
  WR1: SIZEOF (QUERY (ac <* application_context |
       NOT (SIZEOF (QUERY (apd <* application_protocol_definition | ac :=:
apd.application)) = 1 ))) = 0;
END_RULE; -- application_context_requires_ap_definition
(*
```

Argument definitions:

application_context: specifies the set of all instances of **application_context**.

application_protocol_definition: specifies the set of all instances of **application_protocol_definition**.

Formal proposition:

WR1: there shall not be any instance of **application_context** that is not referred to by exactly one instance of **application_protocol_definition**.

5.2.5.6 approval_requires_approval_date_time

The rule **approval_requires_approval_date_time** specifies that each instance of **approval** shall be referred to by exactly one instance of **approval_date_time**.

EXPRESS specification:

```
*)
RULE approval_requires_approval_date_time FOR (approval,
         approval_date_time);
WHERE
  WR1: SIZEOF (QUERY ( app <* approval |
       NOT (SIZEOF (QUERY (adt <* approval_date_time |
       app :=: adt.dated_approval)) = 1))) = 0;
END_RULE; -- approval_requires_approval_date_time
(*
```

Argument definitions:

approval: specifies the set of all instances of **approval**.

approval_date_time: specifies the set of all instances of **approval_date_time**.

Formal proposition:

WR1: there shall not be any instance of **approval** that is not referred to by exactly one **approval_date_time**.

5.2.5.7 approval_date_time_requires_date_time

The rule **approval_date_time_requires_date_time** specifies that each instance of **approval_date_time** shall refer to an instance of date_and_time.

EXPRESS specification:

```
*)
RULE approval_date_time_requires_date_time FOR (approval_date_time);
WHERE
  WR1: SIZEOF (QUERY ( app <* approval_date_time |
       NOT ('TAS_SCHEMA.DATE_AND_TIME' IN TYPEOF(app.date_time)))) = 0;
END_RULE;
(*
```

Argument definitions:

approval_date_time: specifies the set of all instances of **approval_date_time**.

Formal proposition:

WR1: there shall not be any instance of **approval_date_time** that does not refer to a **date_and_time**.

5.2.5.8 approval_requires_approval_person_organization

The rule **approval_requires_approval_person_organization** specifies that each instance of **approval** shall be referred to by exactly one instance of **approval_person_organization**.

EXPRESS specification:

```
*)
RULE approval_requires_approval_person_organization FOR (approval,
         approval_person_organization);
WHERE
  WR1: SIZEOF (QUERY (app <* approval |
       NOT (SIZEOF (QUERY (apo <* approval_person_organization |
       app :=: apo.authorized_approval)) = 1))) = 0;
END_RULE; -- approval_requires_approval_person_organization
(*
```

Argument definitions:

approval: specifies the set of all instances of **approval**.

approval_person_organization: specifies the set of all instances of **approval_person_organization**.

Formal proposition:

WR1: there shall not be any instance of **approval** that is not referred to by exactly one **approval_person_organization**.

5.2.5.9 approval_person_organization_requires_person_organization

The rule **approval_person_organization_requires_person_organization** specifies that each instance of **approval_person_organization** shall refer to an instance of **person_and_organization**.

EXPRESS specification:

```
*)
RULE approval_person_organization_requires_person_organization FOR
(approval_person_organization);
WHERE
  WR1: SIZEOF (QUERY ( app <* approval_person_organization |
      NOT ('TAS_SCHEMA.PERSON_AND_ORGANIZATION' IN TYPEOF(app.person_organization)
))) = 0;
END_RULE;
(*
```

Argument definitions:

approval_person_organization: specifies the set of all instances of **approval_person_organization**.

Formal proposition:

WR1: there shall not be any instance of **approval_person_organization** that does not refer to a **person_and_organization**.

5.2.5.10 approvals_are_assigned

The rule **approvals_are_assigned** specifies that each instance of **approval** shall be referred to at least one instance of **TAS_approval_assignment** that relates it to the approved objects.

EXPRESS specification:

```
*)
RULE approvals_are_assigned FOR
  (approval, TAS_approval_assignment);
WHERE
  WR1: SIZEOF (QUERY (app <* approval |
       NOT (SIZEOF (QUERY (aa <* TAS_approval_assignment |
       app :=: aa.assigned_approval )) >= 1 ))) = 0;
END_RULE; -- approvals_are_assigned
(*
```

Argument definitions:

approval: specifies the set of all instances of **approval**.

TAS_approval_assignment: specifies the set of all instances of **TAS_approval_assignment**.

Formal proposition:

WR1: there shall not be any instance of **approval** that is not referred to by at least one **TAS_approval_assignment**.

5.2.5.11 component_sequence_requires_aspect

The rule **component_sequence_requires_aspect** specifies that each instance of **SIR_component_sequence** shall be referred to by at least one **SIR_aspect**.

EXPRESS specification:

```
*)
RULE component_sequence_requires_aspect FOR (SIR_component_sequence);
WHERE
   WR1: SIZEOF(QUERY(scs <* SIR_component_sequence | SIZEOF(USEDIN(scs,
'TAS_SCHEMA.SIR_ASPECT.COMPONENT_SEQUENCE’)) =0 )) =0;
END_RULE;
(*
```

Argument definitions:

SIR_component_sequence: specifies the set of all instances of **SIR_component_sequence**.

Formal proposition:

WR1: any **SIR_component_sequence** shall be referred to by at least one **SIR_aspect**.

5.2.5.12 dependent_instantiable_approval_role

The rule **dependent_instantiable_approval_role** specifies that each instance of **approval_role** shall be referred to by at least another instance.

EXPRESS specification:

```
*)
RULE dependent_instantiable_approval_role FOR (approval_role);
WHERE
  WR1: SIZEOF (QUERY (dtr <* approval_role |
       NOT (SIZEOF (USEDIN (dtr, '')) >= 1))) = 0;
END_RULE;
(*
```

Argument definitions:

approval_role: specifies the set of all instances of **approval_role**.

Formal proposition:

WR1: there shall not be any instance of **approval_role** that is not referred to by at least one instance.

5.2.5.13 dependent_instantiable_approval_status

The rule **dependent_instantiable_approval_status** specifies that each instance of **approval_status** shall be referred to by at least another instance.

EXPRESS specification:

```
*)
RULE dependent_instantiable_approval_status FOR (approval_status);
WHERE
WR1: SIZEOF (QUERY (ast <* approval_status |
     NOT (SIZEOF (USEDIN (ast, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_approval_status
(*
```

Argument definitions:

approval_status: specifies the set of all instances of **approval_status**.

Formal proposition:

WR1: any instance of **approval_statu**s shall be referred to by at least one instance.

5.2.5.14 dependent_instantiable_date

The rule **dependent_instantiable_date** specifies that each instance of **date** shall be referred to by at least an instance of **date_and_time**.

EXPRESS specification:

```
*)
RULE dependent_instantiable_date FOR (date);
WHERE
   WR1: SIZEOF (QUERY (dt <* date |
       NOT (SIZEOF(USEDIN (dt, 'TAS_SCHEMA.DATE_AND_TIME.DATE_COMPONENT')) >= 1)))
= 0;
END_RULE; -- dependent_instantiable_date
(*
```

Argument definitions:

date: specifies the set of all instances of **date**.

Formal proposition:

WR1: there shall not be any instance of **date** that is not referred to by at least one instance of **date_and_time**.

5.2.5.15 dependent_instantiable_date_time_role

The rule **dependent_instantiable_date_time_role** specifies that each instance of **date_time_role** shall be referred to by at least another instance.

EXPRESS specification:

```
*)
RULE dependent_instantiable_date_time_role FOR (date_time_role);
WHERE
  WR1: SIZEOF (QUERY (dtr <* date_time_role |
       NOT (SIZEOF (USEDIN (dtr, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_date_time_role
(*
```

Argument definitions:

date_time_role: specifies the set of all instances of **date_time_role**.

Formal proposition:

WR1: there shall not be any instance of **date_time_role** that is not referred to by at least one instance.

5.2.5.16 dependent_instantiable_person

The rule **dependent_instantiable_person** specifies that each instance of **person** shall be referred to by at least an instance of **person_and_organization**.

EXPRESS specification:

```
*)
RULE dependent_instantiable_person FOR (person);
WHERE
   WR1: SIZEOF(QUERY(pers <* person |
SIZEOF(USEDIN(pers,'TAS_SCHEMA.PERSON_AND_ORGANIZATION.THE_PERSON’))<1)) = 0;
END_RULE;
(*
```

Argument definition:

person: specifies the set of all instances of **person**.

Formal proposition:

WR1: any instance of **person** shall be referred to by at least one **person_and_organization**.

5.2.5.17 dependent_instantiable_person_and_organization_role

The rule **dependent_instantiable_person_and_organization_role** specifies that each instance of **person_and_organization_role** shall be referred to by at least another instance.

EXPRESS specification:

```
*)
RULE dependent_instantiable_person_and_organization_role FOR (
         person_and_organization_role);
WHERE
  WR1: SIZEOF (QUERY (poar <* person_and_organization_role |
       NOT (SIZEOF (USEDIN (poar, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_person_and_organization_role
(*
```

Argument definitions:

person_and_organization_role: specifies the set of all instances of **person_and_organization_role**.

Formal proposition:

WR1: there shall not be any instance of **person_and_organization_role** that is not referred to by at least one instance.

5.2.5.18 dependent_instantiable_security_classification_level

The rule **dependent_instantiable_security_classification_level** specifies that each instance of **security_classification_level** shall be referred to by at least another instance.

EXPRESS specification:

```
*)
RULE dependent_instantiable_security_classification_level FOR
(security_classification_level);
WHERE
  WR1: SIZEOF (QUERY (scl <* security_classification_level |
       NOT (SIZEOF (USEDIN (scl, '')) >= 1))) = 0;
END_RULE; -- dependent_instantiable_security_classification_level
(*
```

Argument definitions:

security_classification_level: specifies the set of all instances of **security_classification_level**.

Formal proposition:

WR1: there shall not be any instance of **security_classification_level** that is not referred to by at least one instance.

5.2.5.19 dependent_instantiable_type_qualifier

The rule **dependent_instantiable_type_qualifier** specifies that each instance of **type_qualifier** shall be referred to by at least another instance.

EXPRESS specification:

```
*)
RULE dependent_instantiable_type_qualifier FOR (type_qualifier);
WHERE
  WR1: SIZEOF (QUERY (poar <* type_qualifier|
       NOT (SIZEOF (USEDIN (poar, '')) >= 1))) = 0;
END_RULE;
(*
```

Argument definitions:

type_qualifier: specifies the set of all instances of **type_qualifier**.

Formal proposition:

WR1: there shall not be any instance of **type_qualifier** that is not referred to by at least one instance.

5.2.5.20 exclusive_subtypes_action_assignment

The rule **exclusive_subtypes_action_assignment** specifies that any instance of **action_assignment** shall be either a **NRF_run_phase**, a **NRF_run_results**, a **NRF_run_sequence_case**, a **NRF_derivation_result** or a **NRF_derivation_bounds**.

EXPRESS specification:

```
*)
RULE exclusive_subtypes_action_assignment FOR (action_assignment);
WHERE
   WR1: SIZEOF (QUERY (act <* action_assignment |
SIZEOF(['TAS_SCHEMA.NRF_RUN_PHASE','TAS_SCHEMA.NRF_RUN_RESULTS',’TAS_SCHEMA.NRF_RU
N_SEQUENCE_CASE’, ‘TAS_SCHEMA.NRF_DERIVATION_RESULT’,
‘TAS_SCHEMA.NRF_DERIVATION_BOUNDS’]*TYPEOF(act))<>1)) = 0;
END_RULE;
(*
```

Argument definitions:

action_assignment: specifies the set of all instances of **action_assignment**.

Formal proposition:

WR1: there shall not be any instance of **action_assignment** that is not an instance either of **NRF_run_phase** or of **NRF_run_results** or of **NRF_run_sequence_case** or of **NRF_derivation_result** or of **NRF_derivation_bounds**.

5.2.5.21 product_requires_version

The rule **product_requires_version** specifies that any instance of **product** shall be referred to by at least one **SIR_product_version**.

EXPRESS specification:

```
*)
RULE product_requires_version FOR (product, SIR_product_version);
WHERE
  WR1: SIZEOF (QUERY (prod <* product |
       NOT (SIZEOF (QUERY (pdf <* SIR_product_version |
       prod :=: pdf\product_definition_formation.of_product )) >= 1 ))) = 0;
END_RULE; -- product_requires_version
(*
```

Argument definitions:

product: specifies the set of all instances of **product**.

SIR_product_version: specifies the set of all instances of **SIR_product_version**.

Formal proposition:

WR1: there shall not be any instance of **product** that is not referred to by at least one **SIR_product_version**.

5.2.5.22 model_requires_definition

The rule **model_requires_definition** specifies that any instance of **SIR_model** shall be referred to by at least one **TAS_model_product_relationship**.

EXPRESS specification:

```
*)
RULE model_requires_definition FOR (SIR_model, TAS_model_product_relationship);
WHERE
   WR1: SIZEOF(QUERY(sm <* SIR_model | SIZEOF(QUERY(nmpr <*
TAS_model_product_relationship | sm :=:
nmpr\property_definition_representation.used_representation))=0))=0;
END_RULE;
(*
```

Argument definitions:

SIR_model: specifies the set of all instances of **SIR_model**.

TAS_model_product_relationship: specifies the set of all instances of **TAS_model_product_relationship**.

Formal proposition:

WR1: there shall not be any instance of **SIR_model** that is not referred to by at least one instance of **TAS_model_product_relationship**.

5.2.5.23 subtype_mandatory_action

The rule **subtype_mandatory_action** specifies that only instances of the subtypes of **action** are permitted.

EXPRESS specification:

```
*)
RULE subtype_mandatory_action FOR (action);
WHERE
   WR1: SIZEOF (QUERY (act <* action |
SIZEOF(['TAS_SCHEMA.NRF_RUN',’TAS_SCHEMA.NRF_RUN_SEQUENCE’,
‘TAS_SCHEMA.NRF_DERIVATION’]*TYPEOF(act))<>1)) = 0;
END_RULE; -- subtype_mandatory_action
(*
```

Argument definitions:

action: specifies the set of all instances of **action**.

Formal proposition:

WR1: there shall not be any instance of **action** that is not an instance either of **NRF_run** or **NRF_run_sequence** or of **NRF_derivation**.

5.2.5.24 subtype_mandatory_action_relationship

The rule **subtype_mandatory_action_relationship** specifies that only instances of the subtypes of **action_relationship** are permitted.

EXPRESS specification:

```
*)
RULE subtype_mandatory_action_relationship FOR (action_relationship);
WHERE
   WR1: SIZEOF (QUERY (act <* action_relationship |
NOT ('TAS_SCHEMA.NRF_RUN_IN_SEQUENCE' IN TYPEOF(act)))) = 0;
```

```
END_RULE;
(*
```

Argument definitions:

action_relationship: specifies the set of all instances of **action_relationship**.

Formal proposition:

WR1: there shall not be any instance of **action_relationship** that is not an instance of **NRF_run_in_sequence**.

5.2.5.25 subtype_mandatory_date

The rule **subtype_mandatory_date** specifies that only instances of the subtype **calendar_date** of **date** are permitted.

EXPRESS specification:

```
*)
RULE subtype_mandatory_date FOR (date);
WHERE
   WR1: SIZEOF (QUERY (act <* date|NOT('TAS_SCHEMA.CALENDAR_DATE' IN TYPEOF(act)))
) = 0;
END_RULE;
(*
```

Argument definitions:

date: specifies the set of all instances of **date**.

Formal proposition:

WR1: there shall not be any instance of **date** that is not an instance of **calendar_date**.

5.2.5.26 subtype_mandatory_product_definition_formation

The rule **subtype_mandatory_product_definition_formation** specifies that only instances of the subtype **SIR_product_version** of **product_definition_formation** are permitted.

EXPRESS specification:

```
*)
RULE subtype_mandatory_product_definition_formation FOR
(product_definition_formation);
WHERE
   WR1: SIZEOF(QUERY(obj<*
product_definition_formation|NOT('TAS_SCHEMA.SIR_PRODUCT_VERSION' IN TYPEOF(obj)))
)= 0;
END_RULE;
(*
```

Argument definitions:

product_definition_formation: specifies the set of all instances of **product_definition_formation**.

Formal proposition:

WR1: there shall not be any instance of **product_definition_formation** that is not an instance of **SIR_product_version**.

5.2.5.27 subtype_mandatory_product_definition_relationship

The rule **subtype_mandatory_product_definition_relationship** specifies that only instances of its subtypes **next_assembly_usage_occurrence** and **TAS_albedo_reflector** are permitted.

EXPRESS specification:

```
*)
RULE subtype_mandatory_product_definition_relationship FOR
(product_definition_relationship);
```

```
WHERE
   WR1: SIZEOF(QUERY(obj<* product_definition_relationship|
SIZEOF('TAS_SCHEMA.NEXT_ASSEMBLY_USAGE_OCCURENCE',
'TAS_SCHEMA.TAS_ALBEDO_REFLECTOR'] * TYPEOF(obj)) <>1))= 0;
END_RULE;
(*
```

Argument definitions:

product_definition_relationship: specifies the set of all instances of **product_definition_relationship**.

Formal proposition:

WR1: there shall not be any instance of **product_definition_relationship** that is not an instance of **next_assembly_usage_occurence** or of **TAS_albedo_reflector**.

5.2.5.28 unique_ids_in_model

The rule **unique_ids_in_model** specifies that in a **SIR_model**, the values of the attribute id of **SIR_node**s, respectively **SIR_node_relationship**s, respectively **SIR_submodel_usage**s shall be unique for each category of entities.

EXPRESS specification:

```
*)
RULE unique_ids_in_model FOR (SIR_model, SIR_node, SIR_node_relationship,
SIR_submodel_usage);
WHERE
   WR1:SIZEOF(QUERY(sm <* SIR_model | NOT (unique_ids(QUERY(sn<* SIR_node | sn IN
sm\representation.items)) AND unique_ids(QUERY(snr<* SIR_node_relationship | snr
IN sm\representation.items)) AND unique_ids(QUERY(su<* SIR_submodel_usage | su IN
sm\representation.items))) )) =0;
END_RULE;
(*
```

Argument definitions:

SIR_model: specifies the set of all instances of **SIR_model**.

SIR_node: specifies the set of all instances of **SIR_node**.

SIR_node_relationship: specifies the set of all instances of **SIR_node_relationship**.

SIR_submodel_usage: specifies the set of all instances of **SIR_submodel_usage**.

Formal propositions:

WR1: there shall not any instance of **SIR_model** such that the identifiers of the **SIR_node** (resp. **SIR_node_relationship**, resp. **SIR_submodel_usage**) that are elements of the **SIR_model** are not unique.

5.2.5.29 product_context_requires_product

The rule **product_context_requires_product** checks that any instance of **product_context** is referred to by at least one instance of **product**.

EXPRESS specification :

```
*)
RULE product_context_requires_product FOR (product_context, product);
WHERE
   wr1: SIZEOF( QUERY(temp <* product_context | (SIZEOF (QUERY
        (prod <* product | temp IN prod.frame_of_reference)) = 0))) = 0;
END_RULE;
(*
```

Argument definition:

product_context: identifies the set of all instances of **product_context**.

product: identifies the set of all instances of **product**.

Formal proposition:

wr1: there shall be no instance of **product_context** which is not referred to by any instance of **product**.

5.2.5.30 prod_def_context_requires_prod_def

The rule **prod_def_context_requires_prod_def** checks that any instance of **product__definition_context** is referred to by at least one instance of **product_definition**.

EXPRESS specification :

```
*)
RULE prod_def_context_requires_prod_def FOR (product_definition_context,
product_definition);
WHERE
   wr1: SIZEOF( QUERY(temp <* product_definition_context | (SIZEOF (QUERY
(def <* product_definition | temp :=: def.frame_of_reference)) = 0))) = 0;
END_RULE;
(*
```

Argument definition:

product_definition_context: identifies the set of all instances of **product_definition_context**.

product_definition: identifies the set of all instances of **product_definition**.

Formal proposition:

wr1: there shall be no instance of **product_definition_context** which is not referred to by any instance of **product_definition**.

5.2.5.31 restrict_boolean_operator

The rule **restrict_boolean_operator** specifies that only the operator difference is allowed in an instance of **SIR_boolean_result_surface**.

EXPRESS specification :

```
*)
RULE restrict_boolean_operator FOR (SIR_boolean_result_surface);
WHERE
   wr1: SIZEOF( QUERY(temp <* SIR_boolean_result_surface | temp.operator <>
difference)) = 0;
END_RULE;
(*
```

Argument definition:

SIR_boolean_result_surface: identifies the set of all instances of **SIR_boolean_result_surface**.

Formal proposition:

wr1: there shall not be any instance of **SIR_boolean_result_surface** that its attribute operator is not difference.

5.2.5.32 restrict_mission_aspects

The rule **restrict_mission_aspect** specifies that an instance of **SIR_space_mission_aspect** shall refer through its attribute characterized_objects either to:

- a **TAS_orbit_arc**, or
- a **TAS_thermal_environment**, or
- a **TAS_material_property_environment**, or
- a **SIR_kinematic_articulation**,

- a **SIR_placement_transformation** and a **TAS_body_motion_representation**.

EXPRESS specification :

```
*)
RULE restrict_mission_aspects FOR (SIR_space_mission_aspect);
WHERE
   wr1: SIZEOF( QUERY(temp <* SIR_space_mission_aspect | NOT (
((SIZEOF(temp.characterized_objects) =1) AND
 (SIZEOF(TYPEOF(temp.characterized_objects[1])*['TAS_SCHEMA.TAS_ORBIT_ARC',
'TAS_SCHEMA.TAS_THERMAL_ENVIRONMENT',
'TAS_SCHEMA.TAS_MATERIAL_PROPERTY_ENVIRONMENT',
'TAS_SCHEMA.SIR_KINEMATIC_ARTICULATION']= 1))) OR
((SIZEOF(temp.characterized_objects) =2) AND
(SIZEOF((TYPEOF(temp.characterized_objects[1])
+TYPEOF(temp.characterized_objects[2]))
*['TAS_SCHEMA.SIR_PLACEMENT_TRANSFORMATION',
'TAS_SCHEMA.TAS_BODY_MOTION_REPRESENTATION']= 2))) ))) =0;
END_RULE;
(*
```

Argument definition:

SIR_space_mission_aspect: identifies the set of all instances of **SIR_space_mission_aspect**.

Formal proposition:

wr1: there shall not be any instance of **SIR_space_mission_aspect** such the following condition is not fulfilled:

- characterized_objects has one element and it is a **TAS_orbit_arc** or a **TAS_thermal_environment** or a **TAS_material_property_environment**, or
- characterized_objects has two elements and they are a **SIR_placement_transformation** and a **TAS_body_motion_representation**.

5.2.5.33 subtype_mandatory_data_environment

The rule **subtype_mandatory_data_environment** specifies that only instances of the subtypes of **data_environment** defined in this protocol are permitted.

EXPRESS specification:

```
*)
RULE subtype_mandatory_data_environment FOR (data_environment);
WHERE
   WR1: SIZEOF (QUERY (dat <* data_environment |
SIZEOF(['TAS_SCHEMA.TAS_MATERIAL_PROPERTY_ENVIRONMENT','TAS_SCHEMA.TAS_THERMAL_ENV
IRONMENT']* TYPEOF(dat))<>1)) = 0;
END_RULE; -- subtype_mandatory_data_environment
(*
```

Argument definitions:

data_environment: specifies the set of all instances of **data_environment**.

Formal proposition:

WR1: there shall not be any instance of **data_environment** that is not an instance either of **TAS_material_property_environment** or a **TAS_thermal_environment**.

5.2.5.34 TAS_global_units

The rule **TAS_global_units** checks that any instance of **global_unit_assigned_context** contains assignments of **length_unit** and **plane_angle_unit**.

EXPRESS specification :

```
*)
RULE TAS_global_units FOR (global_unit_assigned_context);
WHERE
wr1: SIZEOF (QUERY( guac <* global_unit_assigned_context |
          NOT (SIZEOF(guac.units) =2 ))) =0;
wr2: SIZEOF (QUERY( guac <* global_unit_assigned_context |
         (SIZEOF(['TAS_SCHEMA.LENGTH_UNIT','TAS_SCHEMA.PLANE_ANGLE_UNIT'] IN
           TYPEOF(guac.units))=0)) =0;
END_RULE;
(*
```

Argument definition:

global_unit_assigned_context: identifies the set of all instances of **global_unit_assigned_context**.

Formal proposition:

wr1: there shall be exactly two units assigned in any instance of **global_unit_assigned_context**;

wr2: there shall be no instance of **global_unit_assigned_context** such that **length_unit** and **plane_angle_unit** are not members of the set of assigned units.

5.2.5.35 TAS_dimensionality_is_3

The rule **TAS_dimensionality_is_3** checks that any instance of **geometric_representation_context** defines a 3D geometrical space.

EXPRESS specification :

```
*)
RULE TAS_dimensionality_is_3 FOR (geometric_representation_context);
WHERE
   wr1: SIZEOF(QUERY (grc <* geometric_representation_context|
          grc.coordinate_space_dimension <> 3)) = 0;
END_RULE;
(*
```

Argument definition:

geometric_representation_context: identifies the set of all instances of **geometric_representation_context**.

Formal proposition:

wr1: there shall be no instance of **geometric_representation_context** such that the dimension_count of the coordinate space is different from 3.

5.2.5.36 TAS_point_id_unique

The rule **TAS_point_id_unique** checks that any instance of **shape_representation** contains **cartesian_point**s having either the name ‘ ’ or such their name is unique in the context of the **shape_representation**.

EXPRESS specification :

```
*)
RULE TAS_point_id_unique FOR (cartesian_point, shape_representation);
WHERE
   wr1: SIZEOF( QUERY( sr <* shape_representation | NOT unique_names(QUERY(cp <*
cartesian_point | (cp\representation_item.name <> ‘ ‘) AND (cp IN
sr\representation.items))) )) =0;
END_RULE;
(*
```

Argument definition:

cartesian_point: identifies the set of all instances of **cartesian_point**;

shape_representation : identifies the set of all instances of **shape_representation**;

Formal proposition:

wr1: there shall be no instance of **cartesian_point** which is an item of the representation of a **TAS_geometric_model** such that this representation contains another instance of **cartesian_point** having the same name.

5.2.5.37 at_most_2_faces_per_surface

The rule **at_most_2_faces_per_surface** checks that any **advanced_face** is referred to by at most two **SIR_thermal_radiative_face_set** and that in the case of two references, the active_side shall be either both **none** or different.

EXPRESS specification :

```
*)
RULE at_most_2_faces_per_surface FOR (SIR_thermal_radiative_face_set,
advanced_face);
WHERE
   wr1: SIZEOF( QUERY( sr <* advanced_face |
(SIZEOF(USEDIN(sr,'TAS_SCHEMA.SIR_THERMAL_RADIATIVE_FACE_SET.BASE_SURFACE'))>2) )
) =0;
   wr2: SIZEOF(QUERY( sr1 <* SIR_thermal_radiative_face_set |
(sr1.active_side<>none) AND
(SIZEOF(QUERY( sr2 <* SIR_thermal_radiative_face_set |
(sr1.base_surface:=:sr2.base_surface) AND (sr1.active_side=sr2.active_side) )) >0)
)) =0;
END_RULE;
(*
```

Argument definition:

SIR_thermal_radiative_face_set: identifies the set of all instances of **SIR_thermal_radiative_face_set**;

advanced_face : identifies the set of all instances of **advanced_face**;

Formal proposition:

wr1: there shall not be any **advanced_face** referred to by more than 2 **SIR_thermal_radiative_face_set**s as base_surface;

wr2: there shall not be two **SIR_thermal_radiative_face_set**s with two identical active_sides except if active_side=none;

5.2.5.38 thermal_face_requires_thermal_model

The rule **thermal_face_requires_thermal_model** specifies that each instance of **SIR_thermal_radiative_face_set** shall be referred to by at least one **TAS_basic_thermal_radiative_model**.

EXPRESS specification:

```
*)
RULE thermal_face_requires_thermal_model FOR (SIR_thermal_radiative_face_set);
WHERE
   WR1: SIZEOF(QUERY(stf <* SIR_thermal_radiative_face_set |
(SIZEOF(QUERY(rep <* using_representations(stf)
|'TAS_SCHEMA.TAS_BASIC_THERMAL_RADIATIVE_MODEL' IN TYPEOF(rep))) =0) )) =0;
END_RULE;
(*
```

Argument definitions:

SIR_thermal_radiative_face_set: specifies the set of all instances of **SIR_thermal_radiative_face_set**.

Formal proposition:

WR1: any **SIR_component_sequence** shall be referred to by at least one **TAS_basic_thermal_radiative_face_set**.

5.2.5.39 kinematic_link_requires_shape

The rule **valid_kinematic_link** specifies that each instance of **SIR_kinematic_link** shall be related to exactly one **kinematic_link_representation_association**. In addition, this **kinematic_link_representation_association** shall refer as rep_2 to a **shape_representation**.

EXPRESS specification:

```
*)
RULE kinematic_link_requires_shape FOR (SIR_kinematic_link);
WHERE
   WR1: SIZEOF(QUERY(skl <* SIR_kinematic_link |

SIZEOF(USEDIN(skl.klrr.geometric_aspects,'TAS_SCHEMA.KINEMATIC_LINK_REPRESENTATION
_ASSOCIATION.REP_1')) <>1 ))= 0;
   WR2: SIZEOF(QUERY(skl <* SIR_kinematic_link |
   SIZEOF(QUERY(klra <*
USEDIN(skl.klrr.geometric_aspects,'TAS_SCHEMA.KINEMATIC_LINK_REPRESENTATION_ASSOCI
ATION.REP_1') | 'TAS_SCHEMA.SHAPE_REPRESENTATION' IN
TYPEOF(klra\representation_relationship.rep_2) ))= 0 )) = 0;
END_RULE;
(*
```

Argument definitions:

SIR_kinematic_link: specifies the set of all instances of **SIR_kinematic_link**.

Formal propositions:

WR1: there shall not be any instance of **SIR_kinematic_link** such it is not indirectly associated with exactly one **kinematic_link_representation_association**;

WR2: there shall not be any instance of **SIR_kinematic_link** such it is not indirectly associated with at least one **kinematic_link_representation_association** that refers to a **shape_representation**

5.2.5.40 restrict_represented_product

The rule **restrict_represented_product** specifies that any instance of **SIR_kinematic_link** shall refer as represented_product either to a **product_definition** or to a **next_assembly_usage_occurrence**. In addition, when represented_product refers to a **product_definition**, this **product_definition** shall not be referred to as next_assembly_usage_occurrence\product_definition_relationship.related_product_definition.

EXPRESS specification:

```
*)
RULE restrict_represented_product FOR (SIR_kinematic_link);
WHERE
   WR1: SIZEOF(QUERY(skl <* SIR_kinematic_link |

SIZEOF(['TAS_SCHEMA.PRODUCT_DEFINITION','TAS_SCHEMA.NEXT_ASSEMBLY_USAGE_OCCURRENCE
']*TYPEOF(skl.represented_product)) <>1 )) = 0;
   WR2: SIZEOF(QUERY(skl <* SIR_kinematic_link |
 ('TAS_SCHEMA.PRODUCT_DEFINITION' IN TYPEOF(skl.represented_product)) AND
 (SIZEOF(USEDIN(skl.represented_product,
'TAS_SCHEMA.PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')) <>0) ))
= 0;
END_RULE;
(*
```

Argument definitions:

SIR_kinematic_link: specifies the set of all instances of **SIR_kinematic_link**.

Formal propositions:

WR1: there shall not be any instance of **SIR_kinematic_link** that does not refer as represented_product to either a **product_definition** or a **next_assembly_usage_occurrence**.

WR2: there shall not be any instance of **SIR_kinematic_link** that refers to a **product_definition** that is referred to as related_product_definition by at least one **product_definition_relationship**.

5.2.5.41 subtype_mandatory_kinematic_link

The rule **subtype_mandatory_kinematic_link** specifies that only instances of the subtype **SIR_kinematic_link** of **kinematic_link** are permitted.

EXPRESS specification:

```
*)
RULE subtype_mandatory_kinematic_link FOR (kinematic_link);
WHERE
   WR1: SIZEOF(QUERY(obj<* kinematic_link | NOT('TAS_SCHEMA.SIR_KINEMATIC_LINK' IN
TYPEOF(obj))))= 0;
END_RULE;
(*
```

Argument definitions:

kinematic_link: specifies the set of all instances of **kinematic_link**.

Formal proposition:

WR1: there shall not be any instance of **kinematic_link** that is not an instance of **SIR_kinematic_link**.

5.2.5.42 mechanism_requires_ground_representation

The rule **mechanism_requires_ground_representation** specifies that any instance of **mechanism** shall be such that SELF.containing_property.ground_definition be associated with a **kinematic_ground_representation** through a **kinematic_property_representation_relation.**

EXPRESS specification:

```
*)
RULE mechanism_requires_ground_representation FOR (mechanism);
WHERE
   WR1: SIZEOF(QUERY(mech <* mechanism |
  SIZEOF(USEDIN(mech.containing_property.ground_definition,
'TAS_SCHEMA.KINEMATIC_PROPERTY_REPRESENTATION_RELATION.DEFINITION')) =0)) = 0;
END_RULE;
(*
```

Argument definitions:

mechanism: specifies the set of all instances of **mechanism**.

Formal propositions:

WR1: there shall not be any instance of **mechanism** such its ground_definition is not referred to by at least one **kinematic_property_representation_relation** as definition.

5.2.6 TAS_schema function definitions

5.2.6.1 get_names

The function **get_names** computes the set of **label**s assigned as a name to a **named_item** through **TAS_name_assignment.**

EXPRESS specification:

```
*)
FUNCTION get_names(prop: named_item): SET OF label;
LOCAL
   i: INTEGER;
   sav: BAG OF TAS_name_assignment;
   result: SET OF label;
END_LOCAL;

sav := USEDIN( prop, 'TAS_SCHEMA.TAS_NAME_ASSIGNMENT.ITEMS');

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
   result[i]:=sav[i].assigned_name;
END_REPEAT;

RETURN(result);
END_FUNCTION;
(*
```

Argument definition:

prop: specifies the **named_item** for which the names are computed.

5.2.6.2 get_tools

The function **get_tools** computes the set of **action_resource**s that refer to a **NRF_run** and having the kind 'tool_or_facility'**.**

EXPRESS specification:

```
*)
FUNCTION get_tools(prop: NRF_run): SET OF action_resource;
LOCAL
   i: INTEGER;
   sav: BAG OF action_resource;
   result: SET OF action_resource;
END_LOCAL;

sav := USEDIN( prop, 'TAS_SCHEMA.ACTION_RESOURCE.USAGE');

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
   IF (sav[i].kind.name='tool_or_facility') THEN
   result[i]:=sav[i];
   END_IF;
END_REPEAT;

RETURN(result);
END_FUNCTION;
(*
```

Argument definition:

prop: specifies the **NRF_run** for which the creating tools and facilities are computed.

5.2.6.3 get_timestamps

The function **get_timestamps** computes the set of **date_and_time**s that are assigned through **TAS_date_assignment** to a **NRF_run** with the role 'start_date_and_time'**.**

EXPRESS specification:

```
*)
```

```
FUNCTION get_timestamps(prop: NRF_run): SET OF date_and_time;
LOCAL
   i: INTEGER;
   sav: BAG OF TAS_date_assignment;
   result: SET OF date_and_time;
END_LOCAL;

sav := USEDIN( prop, 'TAS_SCHEMA.TAS_DATE_ASSIGNMENT.ITEMS');

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
   IF (sav[i].role.name='start_date_and_time') THEN
   result[i]:=sav[i].assigned_date_and_time;
   END_IF;
END_REPEAT;

RETURN(result);
END_FUNCTION;
(*
```

Argument definition:

prop: specifies the **NRF_run** for which the starting dates are computed.

5.2.6.4 unique_ids

The function **unique_ids** checks whether the attributes id of a bag of items are unique**.**

The result is TRUE if the ids are unique.

EXPRESS specification:

```
*)
FUNCTION unique_ids(sav: BAG OF GENERIC): LOGICAL;
LOCAL
   i: INTEGER;
   bag_id: BAG OF identifier;
END_LOCAL;

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
   bag_id[i] := sav[i].id;
END_REPEAT;

RETURN(VALUE_UNIQUE(bag_id));
END_FUNCTION;
(*
```

Argument definition:

sav: specifies the bag of generic items for which the unicity of the ids is checked.

5.2.6.5 unique_names

The function **unique_names** checks whether the attributes name of a bag of items are unique**.**

The result is TRUE if the names are unique.

EXPRESS specification:

```
*)
FUNCTION unique_names(sav: BAG OF GENERIC): LOGICAL;
LOCAL
   i: INTEGER;
   bag_names: BAG OF label;
END_LOCAL;

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
   bag_names[i] := sav[i].name;
END_REPEAT;
```

```
RETURN(VALUE_UNIQUE(bag_names));
END_FUNCTION;
(*
```

Argument definition:

sav: specifies the bag of generic items for which the unicity of the names is checked.

5.2.6.6 valid_properties_of_celestial_body

The function **valid_properties_of_celestial_body** checks if a **TAS_celestial_body** has thermal and gravitational properties. The function return TRUE in such a case, FALSE otherwise.

EXPRESS specification :

```
*)
FUNCTION valid_properties_of_celestial_body(item:TAS_celestial_body):BOOLEAN;
LOCAL
   result: BOOLEAN;
   prod_def_form_set: BAG OF product_definition_formation;
   prod_def_set : BAG OF product_definition;
   prop_def_set: BAG OF property_definition;
   i,j,k: INTEGER;
   nb_prop1, nb_prop2: INTEGER;
END_LOCAL;

prod_def_form_set := QUERY(pdf <* USEDIN(item,
'TAS_SCHEMA.PRODUCT_DEFINITION_FORMATION.OF_PRODUCT'|
('TAS_SCHEMA.SIR_PRODUCT_VERSION' IN TYPEOF(pdf)));

REPEAT k:=LOINDEX(prod_def_form_set) TO HIINDEX(prod_def_form_set);
   prod_def_set:= USEDIN(prod_def_form_set[k],
'TAS_SCHEMA.PRODUCTION_DEFINITION.FORMATION');
   nb_prop1:=0;
   nb_prop2:=0;

(* checks if the expected properties are present *)
   REPEAT i:=LOINDEX(prod_def_set) TO HIINDEX(prod_def_set);
      prop_def_set:= USEDIN(prod_def_set[i],
'TAS_SCHEMA.PROPERTY_DEFINITION.DEFINITION');
      REPEAT j:=LOINDEX(prop_def_set) TO HIINDEX(prop_def_set);
         IF ('TAS_SCHEMA.TAS_GRAVITATIONAL_ASPECTS_OF_CELESTIAL_BODY' IN
TYPEOF(prop_def_set[j])) THEN
            nb_prop1:= nb_prop1 + 1;
         END_IF;
         IF ('TAS_SCHEMA.TAS_THERMAL_ASPECTS_OF_CELESTIAL_BODY' IN
TYPEOF(prop_def_set[j])) THEN
            nb_prop2:= nb_prop2 + 1;
         END_IF;
      END_REPEAT;
   END_REPEAT;

   IF NOT ((nb_prop1=1) AND (nb_prop2<=1)) THEN
      RETURN(FALSE);
   END_IF;
END_REPEAT;

RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

item: specifies a **TAS_celestial_body**. This is the input of the function.

5.2.6.7 orbit_assigned_to

The function **orbit_assigned_to** checks whether a **TAS_orbit_arc** is associated with the considered **TAS_celestial_body_with_orbit**. It returns TRUE in such a case, FALSE otherwise.

EXPRESS specification :

```
*)
FUNCTION orbit_assigned_to(item: TAS_celestial_body_with_orbit):BOOLEAN;
LOCAL
   prod_def_form_set: BAG OF product_definition_formation;
   prod_def_set: BAG OF product_definition;
   prop_def_set: BAG OF property_definition;
   orbit_set: BAG OF TAS_orbit_assignment;
   i,j,k: INTEGER;
   nb_orbit: INTEGER;
END_LOCAL;

prod_def_form_set := QUERY(pdf <* USEDIN(item,
'TAS_SCHEMA.PRODUCT_DEFINITION_FORMATION.OF_PRODUCT') |
('TAS_SCHEMA.SIR_PRODUCT_VERSION' IN TYPEOF(pdf)));

REPEAT i:=LOINDEX(prod_def_form_set) TO HIINDEX(prod_def_form_set);
   prod_def_set:= USEDIN(prod_def_form_set[i],
'TAS_SCHEMA.PRODUCTION_DEFINITION.FORMATION');
   nb_orbit:=0;

   REPEAT j:=LOINDEX(prod_def_set) TO HIINDEX(prod_def_set);
      prop_def_set:= USEDIN(prod_def_set[j],
'TAS_SCHEMA.PROPERTY_DEFINITION.DEFINITION');
      REPEAT k:=LOINDEX(prop_def_set) TO HIINDEX(prop_def_set);
         orbit_set:= USEDIN(prop_def_set[k],
'TAS_SCHEMA.TAS_ORBIT_ASSIGNMENT.RELATED_PROPERTY_DEFINITION');
        nb_orbit:= nb_orbit + SIZEOF(orbit_set);
     END_REPEAT;
   END_REPEAT;

   IF (nb_orbit<>1) THEN
      RETURN(FALSE);
   END_IF;
END_REPEAT;

RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definition:

item: specifies a **TAS_celestial_body_with_orbit** for which the associated **TAS_orbit_arc** is looked for.

5.2.6.8 TAS_valid_associated_kepler_set

The function **TAS_valid_associated_kepler_set** checks if the possible association of the **geom_rep** of an instance of **TAS_discretised_orbit_arc** with kepler parameters is correct.

EXPRESS specification :

```
*)
FUNCTION TAS_valid_associated_kepler_set(item:TAS_discretised_orbit_arc):BOOLEAN;
LOCAL
   result: BOOLEAN;
   auxr: SET OF representation_relationship;
END_LOCAL;

auxr:= USEDIN(item.geom_rep, 'TAS_SCHEMA.REPRESENTATION_RELATIONSHIP.REP_1');
result := (SIZEOF(auxr) <= 1);
```

```
IF result AND (SIZEOF(auxr) = 1) THEN
   result:= (auxr[1].description = 'association with unperturbed orbit data') AND
('TAS_SCHEMA.TAS_SPACE_COORDINATE_SYSTEM' IN TYPEOF
(auxr[1].rep_2.context_of_items)) AND
(SIZEOF(auxr[1].rep_2.items) = 1);
IF result THEN
   result:= ('TAS_SCHEMA.TAS_KEPLER_SET' IN TYPEOF(auxr[1].rep_2.items[1]));
END_IF;
END_IF;

RETURN(result);
END_FUNCTION;
(*
```

Argument definitions:

item: instance of **TAS_discretised_orbit_arc**. This is the input of the function.

5.2.6.9 TAS_valid_albedo_reflector

The function **TAS_valid_albedo_reflector** checks if an instance of **TAS_albedo_reflector** has the expected structure and properties.

EXPRESS specification :

```
*)
FUNCTION TAS_valid_albedo_reflector(item:TAS_albedo_reflector):BOOLEAN;
LOCAL
related: product_definition;
relating: product_definition;
result: BOOLEAN;
auxp, auxp2: BAG OF property_definition;
auxr: BAG OF property_definition_representation;
END_LOCAL;

related:= item\product_definition_relationship.related_product_definition;
relating:= item\product_definition_relationship.relating_product_definition;

result:= 'TAS_SCHEMA.TAS_CELESTIAL_BODY'IN TYPEOF(relating.formation.of_product);

IF (result) THEN
   result := (related :<>: relating);
END_IF;

IF (result) THEN
   auxp:= USEDIN(item, 'TAS_SCHEMA.PROPERTY_DEFINITION.DEFINITION');
   auxp2:= QUERY( prop_def <* auxp | (prop_def.name = 'albedo reflection
characteristics'));
   result := (SIZEOF(auxp2) = 1);
END_IF;

IF (result) THEN
   auxr:= QUERY( prdr <*
USEDIN(auxp2,'TAS_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') |
('TAS_SCHEMA.TAS_ALBEDO_REFLECTION_DATA' IN TYPEOF(prdr)));
   result:= (SIZEOF(auxr) =1);
END_IF;

RETURN(result);
END_FUNCTION;
(*
```

Argument definitions:

item: instance of **TAS_albedo_reflector**. This is the input of the function.

```
*)
END_SCHEMA; -- TAS_schema
(*
```

Clause 6 - Conformance Requirements

Conformance to this application protocol includes satisfying the requirements stated in this document, the requirements of the implementation method(s) supported and the relevant requirements of the normative references.

An implementation shall support at least one of the following implementation methods:

- ISO 10303-21.

Requirements with respect to implementation methods are specified in annex C.

The Protocol Information conformance Statement (PICS) proforma lists

This application protocol provides for a number of options that may be supported by an implementation. These options have been grouped into the following conformance classes:

- CC-1: thermal radiation model with basic geometry;
- CC-2: thermal radiation model with basic geometry and kinematic model;
- CC-3: thermal radiation model with basic and constructive geometry;
- CC-4: thermal radiation model with basic and constructive geometry and kinematic model;
- CC-5: thermal radiation model with basic geometry, kinematic model and space mission aspects;
- CC-6: thermal radiation model with basic and constructive geometry, kinematic model and space mission aspects.

Support for a particular conformance class requires support of all the options specified in this class.

Table 6-1 below specifies which AIM elements shall be supported for each of the conformance classes.

Note 98 – The AIM elements are listed in the order in which they appear in the ISO 10303 and 13584 Integrated Resources and Space-domain Integrated Resources.

Note 99 – Some AIM elements are not required in any conformance class. These elements appear in the list, because they are part of the AICs [STEP-509] and/or [STEP-511]. They are however not required in the STEP-TAS ARM and therefore not yet supported in the STEP-TAS protocol. To achieve in the future full compatibility with [STEP-203] conformance class 4 support for these elements will become necessary.

Table 6–1 - Required AIM elements per conformance class

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
application_context	X	X	X	X	X	X
application_protocol_definition	X	X	X	X	X	X
application_context_element	X	X	X	X	X	X
product_context	X	X	X	X	X	X
product_definition_context	X	X	X	X	X	X
product	X	X	X	X	X	X
product_category					X	X
product_related_product_category					X	X
product_definition_formation	X	X	X	X	X	X
product_definition	X	X	X	X	X	X
product_definition_relationship	X	X	X	X	X	X
characterized_object	X	X	X	X	X	X
property_definition	X	X	X	X	X	X
product_definition_shape	X	X	X	X	X	X
shape_aspect	X	X	X	X	X	X
shape_representation	X	X	X	X	X	X
property_definition_representation	X	X	X	X	X	X
name_assignment	X	X	X	X	X	X
document_reference	X	X	X	X	X	X
action_assignment					X	X
approval_assignment	X	X	X	X	X	X
security_classification_assignment	X	X	X	X	X	X
organization_assignment	X	X	X	X	X	X
person_and_organization_assignment	X	X	X	X	X	X
date_and_time_assignment	X	X	X	X	X	X
group_assignment	X	X	X	X	X	X
document_type	X	X	X	X	X	X
document	X	X	X	X	X	X
document_usage_constraint	X	X	X	X	X	X
action					X	X
executed_action					X	X
action_relationship					X	X
action_method					X	X
action_resource					X	X
action_resource_type					X	X
approval_status	X	X	X	X	X	X
approval	X	X	X	X	X	X
approval_date_time	X	X	X	X	X	X
approval_person_organization	X	X	X	X	X	X

Table 6–1 - Required AIM elements per conformance class (Continued)

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
approval_role	X	X	X	X	X	X
security_classification_level	X	X	X	X	X	X
security_classification	X	X	X	X	X	X
address	X	X	X	X	X	X
personal_address	X	X	X	X	X	X
organizational_address	X	X	X	X	X	X
person	X	X	X	X	X	X
organization	X	X	X	X	X	X
organizational_project	X	X	X	X	X	X
person_and_organization	X	X	X	X	X	X
person_and_organization_role	X	X	X	X	X	X
organization_role	X	X	X	X	X	X
date	X	X	X	X	X	X
calendar_date	X	X	X	X	X	X
coordinated_universal_time_offset	X	X	X	X	X	X
local_time	X	X	X	X	X	X
date_and_time	X	X	X	X	X	X
date_time_role	X	X	X	X	X	X
group	X	X	X	X	X	X
effectivity					X	X
named_unit	X	X	X	X	X	X
si_unit	X	X	X	X	X	X
conversion_based_unit	X	X	X	X	X	X
context_dependent_unit	X	X	X	X	X	X
length_unit	X	X	X	X	X	X
mass_unit	X	X	X	X	X	X
time_unit	X	X	X	X	X	X
electric_current_unit	X	X	X	X	X	X
thermodynamic_temperature_unit	X	X	X	X	X	X
amount_of_substance_unit	X	X	X	X	X	X
luminous_intensity_unit	X	X	X	X	X	X
plane_angle_unit	X	X	X	X	X	X
solid_angle_unit	X	X	X	X	X	X
area_unit	X	X	X	X	X	X
volume_unit	X	X	X	X	X	X
ratio_unit	X	X	X	X	X	X
dimensional_exponents	X	X	X	X	X	X
derived_unit_element	X	X	X	X	X	X
derived_unit	X	X	X	X	X	X
global_unit_assigned_context	X	X	X	X	X	X

Table 6–1 - Required AIM elements per conformance class (Continued)

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
measure_with_unit					X	X
length_measure_with_unit					X	X
mass_measure_with_unit					X	X
time_measure_with_unit					X	X
electric_current_measure_with_unit					X	X
thermodynamic_temperature_measure_with_unit					X	X
amount_of_substance_measure_with_unit					X	X
luminous_intensity_measure_with_unit					X	X
plane_angle_measure_with_unit					X	X
solid_angle_measure_with_unit					X	X
area_measure_with_unit					X	X
volume_measure_with_unit					X	X
ratio_measure_with_unit					X	X
geometric_representation_context	X	X	X	X	X	X
geometric_representation_item	X	X	X	X	X	X
point	X	X	X	X	X	X
cartesian_point	X	X	X	X	X	X
point_on_curve						
point_on_surface						
degenerate_pcurve						
evaluated_degenerate_pcurve						
direction	X	X	X	X	X	X
vector	X	X	X	X	X	X
placement	X	X	X	X	X	X
axis1_placement	X	X	X	X	X	X
axis2_placement_2d						
axis2_placement_3d	X	X	X	X	X	X
cartesian_transformation_operator	X	X	X	X	X	X
cartesian_transformation_operator_3d	X	X	X	X	X	X
curve	X	X	X	X	X	X
line	X	X	X	X	X	X
conic	X	X	X	X	X	X
circle	X	X	X	X	X	X
ellipse						
hyperbola						
parabola	X	X	X	X	X	X
bounded_curve	X	X	X	X	X	X
polyline						
b_spline_curve						
b_spline_curve_with_knots						

Table 6–1 - Required AIM elements per conformance class (Continued)

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
uniform_curve						
quasi_uniform_curve						
bezier_curve						
rational_b_spline_curve						
composite_curve						
composite_curve_segment						
pcurve						
surface_curve						
intersection_curve						
seam_curve						
bounded_surface_curve						
offset_curve_3d						
curve_replica						
surface	X	X	X	X	X	X
elementary_surface	X	X	X	X	X	X
plane	X	X	X	X	X	X
cylindrical_surface	X	X	X	X	X	X
conical_surface	X	X	X	X	X	X
spherical_surface	X	X	X	X	X	X
toroidal_surface						
degenerate_toroidal_surface						
swept_surface	X	X	X	X	X	X
surface_of_linear_extrusion						
surface_of_revolution	X	X	X	X	X	X
bounded_surface	X	X	X	X	X	X
b_spline_surface						
b_spline_surface_with_knots						
uniform_surface						
quasi_uniform_surface						
bezier_surface						
rational_b_spline_surface						
offset_surface						
surface_replica	X	X	X	X	X	X
topological_representation_item	X	X	X	X	X	X
vertex	X	X	X	X	X	X
vertex_point	X	X	X	X	X	X
edge	X	X	X	X	X	X
edge_curve	X	X	X	X	X	X
oriented_edge	X	X	X	X	X	X
path	X	X	X	X	X	X

Table 6–1 - Required AIM elements per conformance class (Continued)

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
oriented_path	X	X	X	X	X	X
loop	X	X	X	X	X	X
vertex_loop						
edge_loop	X	X	X	X	X	X
face_bound	X	X	X	X	X	X
face_outer_bound	X	X	X	X	X	X
face	X	X	X	X	X	X
face_surface	X	X	X	X	X	X
oriented_face	X	X	X	X	X	X
connected_face_set	X	X	X	X	X	X
open_shell	X	X	X	X	X	X
oriented_open_shell	X	X	X	X	X	X
closed_shell	X	X	X	X	X	X
oriented_closed_shell	X	X	X	X	X	X
shell_based_surface_model	X	X	X	X	X	X
half_space_solid			X	X		X
uncertainty_measure_with_unit	X	X	X	X	X	X
representation_context	X	X	X	X	X	X
global_uncertainty_assigned_context	X	X	X	X	X	X
representation_item	X	X	X	X	X	X
representation	X	X	X	X	X	X
representation_relationship		X		X	X	X
item_defined_transformation		X		X	X	X
functionally_defined_transformation	X	X	X	X	X	X
representation_map	X	X	X	X	X	X
mapped_item	X	X	X	X	X	X
definitional_representation						
parametric_representation_context						
product_definition_usage	X	X	X	X	X	X
assembly_component_usage	X	X	X	X	X	X
next_assembly_usage_occurrence	X	X	X	X	X	X
colour	X	X	X	X	X	X
colour_specification	X	X	X	X	X	X
colour_rgb	X	X	X	X	X	X
material_property	X	X	X	X	X	X
property_definition_relationship					X	X
material_designation	X	X	X	X	X	X
material_designation_characterization	X	X	X	X	X	X
material_property_representation	X	X	X	X	X	X
data_environment	X	X	X	X	X	X

Table 6–1 - Required AIM elements per conformance class (Continued)

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
type_qualifier	X	X	X	X	X	X
measure_representation_item					X	X
descriptive_representation_item					X	X
rotation_about_direction		X		X	X	X
kinematic_property_definition		X		X	X	X
kinematic_property_representation_relation		X		X	X	X
kinematic_ground_representation		X		X	X	X
mechanism		X		X	X	X
kinematic_structure		X		X	X	X
kinematic_joint		X		X	X	X
kinematic_link		X		X	X	X
kinematic_link_representation_relation		X		X	X	X
kinematic_link_representation		X		X	X	X
kinematic_link_representation_association		X		X	X	X
kinematic_pair		X		X	X	X
pair_actuator		X		X	X	X
pair_value		X		X	X	X
simple_pair_range		X		X	X	X
revolute_pair		X		X	X	X
revolute_pair_range		X		X	X	X
revolute_pair_value		X		X	X	X
prismatic_pair		X		X	X	X
prismatic_pair_range		X		X	X	X
prismatic_pair_value		X		X	X	X
translation					X	X
generic_expression					X	X
simple_gen_expression					X	X
generic_literal					X	X
generic_variable					X	X
variable_semantics					X	X
environment					X	X
unary_gen_expression					X	X
binary_gen_expression					X	X
m_ary_gen_expression					X	X
expression					X	X
variable					X	X
numeric_expression					X	X
simple_n_expression					X	X
literal_number					X	X
int_literal					X	X

Table 6–1 - Required AIM elements per conformance class (Continued)

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
real_literal					X	X
numeric_variable					X	X
int_numeric_variable					X	X
real_numeric_variable					X	X
unary_n_expression					X	X
binary_n_expression					X	X
m_ary_n_expression					X	X
plus_expression					X	X
mult_expression					X	X
power_expression					X	X
boolean_expression					X	X
unary_B_expression					X	X
binary_B_expression					X	X
m_ary_B_expression					X	X
comparison_expression					X	X
comparison_greater_equal					X	X
comparison_less_equal					X	X
interval_expression					X	X
mathematical_string					X	X
SIR_product_version	X	X	X	X	X	X
SIR_variable					X	X
SIR_parameterized_function					X	X
SIR_polynomial_function					X	X
SIR_tabular_function					X	X
SIR_cyclic_tabular_function					X	X
SIR_model	X	X	X	X	X	X
SIR_node	X	X	X	X	X	X
SIR_node_relationship	X	X	X	X	X	X
SIR_submodel_usage	X	X	X	X	X	X
SIR_node_usage	X	X	X	X	X	X
SIR_node_relationship_usage	X	X	X	X	X	X
SIR_component_sequence					X	X
SIR_list_of_descriptive_values	X	X	X	X	X	X
SIR_list_of_functions					X	X
SIR_property_name	X	X	X	X	X	X
SIR_property_class	X	X	X	X	X	X
SIR_property_scalar	X	X	X	X	X	X
SIR_property_descriptive	X	X	X	X	X	X
SIR_property_functional					X	X
SIR_applicability_of_values	X	X	X	X	X	X

Table 6–1 - Required AIM elements per conformance class (Continued)

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
SIR_property_quantitative	X	X	X	X	X	X
SIR_property_tensor	X	X	X	X	X	X
SIR_scalar_in_tensor	X	X	X	X	X	X
SIR_property_meaning	X	X	X	X	X	X
SIR_aspect					X	X
SIR_ato_campaign	X	X	X	X	X	X
SIR_ato_case	X	X	X	X	X	X
SIR_case_in_campaign	X	X	X	X	X	X
SIR_ato_phase					X	X
SIR_initial_conditioning					X	X
SIR_property_usage	X	X	X	X	X	X
SIR_scan					X	X
NRF_scan_sampled					X	X
NRF_scan_derived					X	X
NRF_derivation_procedure					X	X
NRF_derivation_result					X	X
NRF_derivation_bounds					X	X
NRF_derivation					X	X
SIR_mask					X	X
SIR_unit_assignment	X	X	X	X	X	X
SIR_unit_with_offset	X	X	X	X	X	X
SIR_number_valued_property	X	X	X	X	X	X
SIR_function_valued_property					X	X
SIR_string_valued_property	X	X	X	X	X	X
SIR_space_mission_case	X	X	X	X	X	X
SIR_space_mission_phase					X	X
SIR_space_aspect_in_phase					X	X
SIR_space_mission_aspect	X	X	X	X	X	X
SIR_space_mission_event					X	X
SIR_event_effectivity					X	X
SIR_event_assignment					X	X
SIR_position_and_velocity					X	X
SIR_kepler_set					X	X
SIR_placement_transformation					X	X
SIR_pointing_in_space					X	X
SIR_space_direction					X	X
SIR_rotation_motion					X	X
SIR_boolean_result_surface			X	X		X
SIR_boolean_result_replica			X	X		X
SIR_thermal_radiative_face_set	X	X	X	X	X	X

Table 6–1 - Required AIM elements per conformance class (Continued)

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
SIR_thermal_radiative_meshing	X	X	X	X	X	X
SIR_thermal_radiative_incremental_meshing	X	X	X	X	X	X
SIR_thermal_radiative_explicit_meshing	X	X	X	X	X	X
SIR_colour_assignment	X	X	X	X	X	X
SIR_implicit_revolute_pair_value					X	X
SIR_kinematic_articulation					X	X
SIR_kinematic_link		X		X	X	X
advanced_face	X	X	X	X	X	X
manifold_surface_shape_representation	X	X	X	X	X	X
TAS_security_assignment	X	X	X	X	X	X
TAS_date_assignment	X	X	X	X	X	X
TAS_approval_assignment	X	X	X	X	X	X
TAS_contact_assignment	X	X	X	X	X	X
TAS_name_assignment	X	X	X	X	X	X
TAS_commercial_designation	X	X	X	X	X	X
TAS_supplier	X	X	X	X	X	X
TAS_colour_assignment	X	X	X	X	X	X
TAS_document_reference	X	X	X	X	X	X
TAS_model_product_relationship	X	X	X	X	X	X
NRF_run					X	X
NRF_run_phase					X	X
NRF_run_results					X	X
NRF_run_sequence					X	X
NRF_run_sequence_case					X	X
NRF_run_in_sequence					X	X
TAS_space_coordinate_system	X	X	X	X	X	X
TAS_gravitational_constant					X	X
TAS_revolution_period					X	X
TAS_notional_distance					X	X
TAS_thermal_flux					X	X
TAS_thermal_radiation_spectrum					X	X
TAS_evaluation_event					X	X
TAS_orbit_governor					X	X
TAS_orbit_assignment					X	X
TAS_orbit_arc					X	X
TAS_discretised_orbit_arc					X	X
TAS_kepler_orbit_arc					X	X
TAS_celestial_body					X	X
TAS_celestial_body_with_orbit					X	X
TAS_gravitational_aspects_of_celestial_body					X	X

Table 6–1 - Required AIM elements per conformance class (Continued)

AIM element	*CC-1*	*CC-2*	*CC-3*	*CC-4*	*CC-5*	*CC-6*
TAS_thermal_aspects_of_celestial_body					X	X
TAS_albedo_reflector					X	X
TAS_albedo_reflection_data					X	X
TAS_body_motion_representation					X	X
TAS_thermal_environment					X	X
TAS_thermal_environment_data					X	X
TAS_material_property_environment	X	X	X	X	X	X
TAS_basic_thermal_radiative_model	X	X	X	X	X	X
TAS_meshed_thermal_radiative_model	X	X	X	X	X	X
TAS_enclosure	X	X	X	X	X	X
TAS_material_property	X	X	X	X	X	X

Annex A - AIM EXPRESS expanded listing (normative)

A.1 Introduction

This file was generated by EXTOOL V3.5, EXPRESS toolkit developed by Association GOSET.

A.2 AIM expanded listing

```
      --  GOSET ASSOCIATION
      --  NANTERRE - FRANCE
      --        .....
      --  EXTOOL : Short to Long Translator
      --  FILE: /home1/certif/general/huau/tas/lform


SCHEMA TAS_schema;
(*
*)


TYPE characterized_definition = SELECT
     (characterized_object,
      characterized_product_definition,
      shape_definition);
   END_TYPE;

TYPE characterized_product_definition = SELECT
     (product_definition,
      product_definition_relationship);
   END_TYPE;

TYPE shape_definition = SELECT
     (product_definition_shape,
      shape_aspect);
   END_TYPE;

TYPE supported_item = SELECT
     (action,
      action_method);
   END_TYPE;

TYPE person_organization_select = SELECT
     (person,
      organization,
      person_and_organization);
   END_TYPE;

TYPE date_time_select = SELECT
     (date,
      local_time,
      date_and_time);
   END_TYPE;

TYPE year_number = INTEGER;
   END_TYPE;

TYPE month_in_year_number = INTEGER;
   WHERE
     WR1: { 1 <= SELF <= 12 };
   END_TYPE;
```

```
TYPE day_in_month_number = INTEGER;
   END_TYPE;

TYPE ahead_or_behind = ENUMERATION OF
      (ahead,
       behind);
   END_TYPE;

TYPE hour_in_day = INTEGER;
   WHERE
     WR1: { 0 <= SELF < 24 };
   END_TYPE;

TYPE minute_in_hour = INTEGER;
   WHERE
     WR1: { 0 <= SELF <= 59 };
   END_TYPE;

TYPE second_in_minute = REAL;
   WHERE
     WR1: { 0 <= SELF < 60 };
   END_TYPE;

TYPE identifier = STRING;
   END_TYPE;

TYPE label = STRING;
   END_TYPE;

TYPE text = STRING;
   END_TYPE;

TYPE measure_value = SELECT
      (length_measure,
       mass_measure,
       time_measure,
       electric_current_measure,
       thermodynamic_temperature_measure,
       amount_of_substance_measure,
       luminous_intensity_measure,
       plane_angle_measure,
       solid_angle_measure,
       area_measure,
       volume_measure,
       ratio_measure,
       parameter_value,
       numeric_measure,
       context_dependent_measure,
       descriptive_measure,
       positive_length_measure,
       positive_plane_angle_measure,
       positive_ratio_measure,
       count_measure);
   END_TYPE;

TYPE length_measure = REAL;
   END_TYPE;

TYPE mass_measure = REAL;
   END_TYPE;

TYPE time_measure = REAL;
   END_TYPE;

TYPE electric_current_measure = REAL;
   END_TYPE;

TYPE thermodynamic_temperature_measure = REAL;
```

```
   END_TYPE;

TYPE amount_of_substance_measure = REAL;
   END_TYPE;

TYPE luminous_intensity_measure = REAL;
   END_TYPE;

TYPE plane_angle_measure = REAL;
   END_TYPE;

TYPE solid_angle_measure = REAL;
   END_TYPE;

TYPE area_measure = REAL;
   END_TYPE;

TYPE volume_measure = REAL;
   END_TYPE;

TYPE ratio_measure = REAL;
   END_TYPE;

TYPE parameter_value = REAL;
   END_TYPE;

TYPE numeric_measure = NUMBER;
   END_TYPE;

TYPE positive_length_measure = length_measure;
   WHERE
     WR1: SELF > 0;
   END_TYPE;

TYPE positive_plane_angle_measure = plane_angle_measure;
   WHERE
     WR1: SELF > 0;
   END_TYPE;

TYPE positive_ratio_measure = ratio_measure;
   WHERE
     WR1: SELF > 0;
   END_TYPE;

TYPE context_dependent_measure = REAL;
   END_TYPE;

TYPE descriptive_measure = STRING;
   END_TYPE;

TYPE count_measure = NUMBER;
   END_TYPE;

TYPE unit = SELECT
     (named_unit,
      derived_unit);
   END_TYPE;

TYPE si_unit_name = ENUMERATION OF
     (metre,
      gram,
      second,
      ampere,
      kelvin,
      mole,
      candela,
      radian,
      steradian,
```

```
      hertz,
      newton,
      pascal,
      joule,
      watt,
      coulomb,
      volt,
      farad,
      ohm,
      siemens,
      weber,
      tesla,
      henry,
      degree_Celsius,
      lumen,
      lux,
      becquerel,
      gray,
      sievert);
   END_TYPE;

TYPE si_prefix = ENUMERATION OF
     (exa,
      peta,
      tera,
      giga,
      mega,
      kilo,
      hecto,
      deca,
      deci,
      centi,
      milli,
      micro,
      nano,
      pico,
      femto,
      atto);
   END_TYPE;

TYPE dimension_count = INTEGER;
WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE transition_code = ENUMERATION OF
   (discontinuous,
    continuous,
    cont_same_gradient,
    cont_same_gradient_same_curvature);
 END_TYPE;

TYPE preferred_surface_curve_representation = ENUMERATION OF
   (curve_3d,
    pcurve_s1,
    pcurve_s2);
 END_TYPE;

TYPE b_spline_curve_form = ENUMERATION OF
   (polyline_form,
    circular_arc,
    elliptic_arc,
    parabolic_arc,
    hyperbolic_arc,
    unspecified);
 END_TYPE;

TYPE b_spline_surface_form = ENUMERATION OF
```

```
   (plane_surf,
    cylindrical_surf,
    conical_surf,
    spherical_surf,
    toroidal_surf,
    surf_of_revolution,
    ruled_surf,
    generalised_cone,
    quadric_surf,
    surf_of_linear_extrusion,
    unspecified);
 END_TYPE;

TYPE knot_type = ENUMERATION OF
   (uniform_knots,
    unspecified,
    quasi_uniform_knots,
    piecewise_bezier_knots);
 END_TYPE;

TYPE axis2_placement = SELECT
   (axis2_placement_2d,
    axis2_placement_3d);
 END_TYPE;

TYPE curve_on_surface = SELECT
   (pcurve,
    surface_curve);
 END_TYPE;

TYPE pcurve_or_surface = SELECT
   (pcurve,
    surface);
 END_TYPE;

TYPE trimming_select = SELECT
   (cartesian_point,
    parameter_value);
 END_TYPE;

TYPE vector_or_direction = SELECT
   (vector,
    direction);
 END_TYPE;

TYPE shell = SELECT
   (open_shell,
    closed_shell);
 END_TYPE;

TYPE reversible_topology_item = SELECT
   (edge,
    path,
    face,
    face_bound,
    closed_shell,
    open_shell);
 END_TYPE;

TYPE list_of_reversible_topology_item =
                                LIST [0:?] of reversible_topology_item;
 END_TYPE;

TYPE set_of_reversible_topology_item =
                        SET [0:?] of reversible_topology_item;
  END_TYPE;

TYPE reversible_topology = SELECT
```

```
            (reversible_topology_item,
             list_of_reversible_topology_item,
             set_of_reversible_topology_item);
 END_TYPE;

TYPE boolean_operator = ENUMERATION OF
   (union,
    intersection,
    difference);
 END_TYPE;

TYPE geometric_set_select = SELECT
   (point,
    curve,
    surface);
 END_TYPE;

TYPE surface_model = SELECT
   (shell_based_surface_model);
 END_TYPE;

TYPE transformation = SELECT
  (item_defined_transformation,
   functionally_defined_transformation);
END_TYPE;

TYPE characterized_material_property = SELECT
  (material_property_representation);
END_TYPE;

TYPE value_qualifier = SELECT
  (type_qualifier);
END_TYPE;

TYPE rigid_placement = SELECT
  (axis2_placement_3d);
END_TYPE;

TYPE rotational_range_measure = SELECT
  (plane_angle_measure,
   unlimited_range);
END_TYPE;

TYPE translational_range_measure = SELECT
  (length_measure,
   unlimited_range);
END_TYPE;

TYPE spatial_rotation = SELECT
  ( rotation_about_direction);
END_TYPE;

TYPE unlimited_range = ENUMERATION OF
  (unlimited);
END_TYPE;

TYPE kinematic_frame_background = SELECT
  (point,
   curve,
   surface);
END_TYPE;

TYPE motion_parameter_measure = SELECT
  (parameter_value,
   measure_with_unit);
END_TYPE;

TYPE less_or_greater = SELECT(
```

```
            comparison_less_equal,
            comparison_greater_equal);
END_TYPE;
TYPE tabular_interpolation = ENUMERATION OF (
            polynomial,
            linear_logarithmic);
END_TYPE;
TYPE tabular_interpolation_degree = INTEGER;
WHERE
      WR1: SELF >= 1;
END_TYPE;
TYPE list_real_literals = LIST[1:?] OF REAL;
END_TYPE;
TYPE list_ascending_real_literals = list_real_literals;
WHERE
      WR1: ascending_list(SELF);
END_TYPE;
TYPE SIR_model_constituent = SELECT(
            SIR_node,
            SIR_node_relationship,
            SIR_submodel_usage
            );
END_TYPE;
TYPE SIR_node_or_usage = SELECT(
            SIR_node,
            SIR_node_usage
            );
END_TYPE;
TYPE SIR_network_component = SELECT(
            SIR_model,
            SIR_node,
            SIR_node_usage,
            SIR_node_relationship,
            SIR_node_relationship_usage,
```

```
            SIR_submodel_usage
            );
END_TYPE;
TYPE SIR_property_symmetry = ENUMERATION OF (symmetrical, antisymmetrical);
END_TYPE;
TYPE SIR_list_of_values_or_functions = SELECT (
            SIR_list_of_descriptive_values, SIR_list_of_functions);
END_TYPE;
TYPE SIR_descriptive_or_functional = SELECT (
            SIR_property_descriptive, SIR_property_functional);
END_TYPE;
TYPE SIR_lifetime_type = ENUMERATION OF (invariant, sample, interval);
END_TYPE ;
TYPE SIR_abscissa_sequencing_type = ENUMERATION OF (
            strictly_decreasing,
            monotonic_decreasing,
            monotonic_increasing,
            strictly_increasing
            );
END_TYPE ;
TYPE SIR_case_or_phase = SELECT (
            SIR_ato_case, SIR_ato_phase);
END_TYPE;
TYPE SIR_number = NUMBER;
END_TYPE;
TYPE SIR_string = STRING;
END_TYPE;
TYPE SIR_number_or_string = SELECT (
            SIR_number, SIR_string);
END_TYPE;
TYPE SIR_index_interval = ARRAY[1:2] OF INTEGER;
WHERE
      WR1: SELF[1] >= 0;
      WR2: SELF[2] >= SELF[1];
```

```
END_TYPE;

TYPE true_anomaly = plane_angle_measure;

END_TYPE;

TYPE event = SELECT ( date_and_time, time_measure, true_anomaly);
END_TYPE;

TYPE considered_aspect = SELECT (
            data_environment,
            SIR_placement_transformation,
            property_definition_representation,
            SIR_kinematic_articulation);
END_TYPE;

TYPE description_or_direction = SELECT (
            descriptive_representation_item,
            direction,
            SIR_space_direction);
END_TYPE;

TYPE spin_positions_number = INTEGER;
WHERE
      WR1: SELF>0;
END_TYPE;

TYPE angular_velocity_or_positions_number = SELECT (
            measure_with_unit,
            spin_positions_number);
WHERE
      WR1: NOT ('TAS_SCHEMA.SPIN_POSITIONS_NUMBER' IN TYPEOF(SELF)) XOR
            valid_velocity(SELF,'PLANE_ANGLE_UNIT');
END_TYPE;

TYPE SIR_boolean_operand_1 = SELECT(
            advanced_face,
            open_shell,
            closed_shell,
            SIR_boolean_result_surface
            );
WHERE
      WR1: (SIZEOF(['TAS_SCHEMA.ADVANCED_FACE',
'TAS_SCHEMA.SIR_BOOLEAN_RESULT_SURFACE']*TYPEOF(SELF))=1) OR
(SIZEOF(QUERY(fac <* SELF\connected_face_set.cfs_faces | NOT
('TAS_SCHEMA.ADVANCED_FACE' IN TYPEOF(fac))))=0);
END_TYPE;

TYPE SIR_boolean_operand_2 = SELECT(
            advanced_face,
            half_space_solid
            );
END_TYPE;

TYPE SIR_activity_side = ENUMERATION OF (
            none,
            positive,
            negative
            );
END_TYPE;

TYPE SIR_meshing_order = ENUMERATION OF (
            direction_1_first,
            direction_2_first
            );
END_TYPE;
```

```
TYPE security_classified_item = SELECT (
      SIR_model,
      SIR_node,
      SIR_node_relationship,
      SIR_aspect,
      SIR_ato_campaign,
      SIR_ato_case,
      SIR_ato_phase
      );
END_TYPE;
(*
*)
TYPE dated_item = SELECT (
      SIR_ato_campaign,
      NRF_run,
      SIR_parameterized_function
      );
END_TYPE;
(*
*)
TYPE approved_item = SELECT (
      SIR_ato_campaign
      );
END_TYPE;
(*
*)
TYPE sourced_item = SELECT (
      SIR_ato_campaign
      );
END_TYPE;
(*
*)
TYPE named_item = SELECT (
      NRF_run,
      NRF_run_sequence,
      mechanism
      );
END_TYPE;
(*
*)
TYPE documented_item = SELECT(
      SIR_parameterized_function);
END_TYPE;
(*
*)
TYPE supplied_item = SELECT(
      material_designation);
END_TYPE;
(*
*)
TYPE coloured_item = SELECT(
      advanced_face);
END_TYPE;
(*
*)
TYPE TAS_radius_of_celestial_body = SELECT (
            measure_representation_item,
            SIR_function_valued_property);
WHERE
      WR1: (('TAS_SCHEMA.MEASURE_REPRESENTATION_ITEM' IN TYPEOF(SELF)) AND
('TAS_SCHEMA.LENGTH_MEASURE_WITH_UNIT' IN TYPEOF(SELF))) XOR
(('TAS_SCHEMA.SIR_FUNCTION_VALUED_PROPERTY' IN TYPEOF(SELF)) AND
(SELF\SIR_function_valued_property.property_usage.meaning.roles[1].name='radius')
AND
(SELF\SIR_function_valued_property.property_usage.property.name.name='length'));
END_TYPE;
(*
*)
```

```
TYPE TAS_black_body_surface_temperature = SELECT (
            measure_representation_item,
            SIR_function_valued_property);
WHERE
      WR1: (('TAS_SCHEMA.MEASURE_REPRESENTATION_ITEM' IN TYPEOF(SELF)) AND
 ('TAS_SCHEMA.THERMODYNAMIC_TEMPERATURE_MEASURE_WITH_UNIT' IN TYPEOF(SELF))) XOR
 (('TAS_SCHEMA.SIR_FUNCTION_VALUED_PROPERTY' IN TYPEOF(SELF)) AND
 (SELF\SIR_function_valued_property.property_usage.meaning.roles[1].name='black
body surface temperature') AND

(SELF\SIR_function_valued_property.property_usage.property.name.name='thermodynami
c temperature'));
END_TYPE;
(*
*)
TYPE TAS_albedo_reflection_factor = SELECT (
            measure_representation_item,
            SIR_function_valued_property);
WHERE
      WR1: (('TAS_SCHEMA.MEASURE_REPRESENTATION_ITEM' IN TYPEOF(SELF)) AND
 ('TAS_SCHEMA.RATIO_MEASURE_WITH_UNIT' IN TYPEOF(SELF))) XOR
(('TAS_SCHEMA.SIR_FUNCTION_VALUED_PROPERTY' IN TYPEOF(SELF)) AND
(SELF\SIR_function_valued_property.property_usage.meaning.roles[1].name='albedo
reflection factor') AND
(SELF\SIR_function_valued_property.property_usage.property.name.name='ratio'));
END_TYPE;
(*
*)
TYPE TAS_property_value = SELECT (
            SIR_number_valued_property,
            SIR_function_valued_property,
            SIR_string_valued_property
            );
END_TYPE;

(*
*)

ENTITY application_context;
     application          : text;
INVERSE
     context_elements : SET [1:?] OF  application_context_element
                                   FOR frame_of_reference;
END_ENTITY;

ENTITY application_protocol_definition;
     status                                    : label;
     application_interpreted_model_schema_name : label;
     application_protocol_year                 : year_number;
     application                               : application_context;
   END_ENTITY;

ENTITY application_context_element
     SUPERTYPE OF (ONEOF (product_context,
                          product_definition_context));
     name                 : label;
     frame_of_reference   : application_context;
   END_ENTITY;

ENTITY product_context
     SUBTYPE OF (application_context_element);
     discipline_type : label;
   END_ENTITY;

ENTITY product_definition_context
     SUBTYPE OF (application_context_element);
     life_cycle_stage : label;
   END_ENTITY;
```

```
ENTITY product;
     id                 : identifier;
     name               : label;
     description        : text;
     frame_of_reference : SET [1:?] OF product_context;
   UNIQUE
     UR1: id;
   END_ENTITY;

ENTITY product_category;
     name        : label;
     description : OPTIONAL text;
   END_ENTITY;

ENTITY product_related_product_category
     SUBTYPE OF (product_category);
     products : SET [1:?] OF product;
   END_ENTITY;

ENTITY product_definition_formation;
     id          : identifier;
     description : text;
     of_product  : product;
   UNIQUE
     UR1: id, of_product;
   END_ENTITY;

ENTITY product_definition;
     id                 : identifier;
     description        : text;
     formation          : product_definition_formation;
     frame_of_reference : product_definition_context;
   END_ENTITY;

ENTITY product_definition_relationship;
     id                          : identifier;
     name                        : label;
     description                 : text;
     relating_product_definition : product_definition;
     related_product_definition  : product_definition;
   END_ENTITY;

ENTITY characterized_object;
     name        : label;
     description : text;
   END_ENTITY;

ENTITY property_definition;
     name        : label;
     description : text;
     definition  : characterized_definition;
   END_ENTITY;

ENTITY product_definition_shape
     SUBTYPE OF (property_definition);
   UNIQUE
     UR1: SELF\property_definition.definition;
   WHERE
     WR1: 'PRODUCT_PROPERTY_DEFINITION.CHARACTERIZED_PRODUCT_DEFINITION'
          IN TYPEOF (SELF\property_definition.definition);
   END_ENTITY;

ENTITY shape_aspect;
     name                : label;
     description         : text;
     of_shape            : product_definition_shape;
     product_definitional : LOGICAL;
```

```
   END_ENTITY;

ENTITY shape_representation
     SUBTYPE OF (representation);
   END_ENTITY;

ENTITY property_definition_representation;
     definition          : property_definition;
     used_representation : representation;
   END_ENTITY;

ENTITY name_assignment
     ABSTRACT SUPERTYPE;
     assigned_name : label;
   END_ENTITY;

ENTITY document_reference
     ABSTRACT SUPERTYPE;
     assigned_document : document;
     source            : label;
   END_ENTITY;

ENTITY action_assignment
     ABSTRACT SUPERTYPE;
     assigned_action : action;
   END_ENTITY;

ENTITY approval_assignment
     ABSTRACT SUPERTYPE;
     assigned_approval : approval;
   END_ENTITY;

ENTITY security_classification_assignment
     ABSTRACT SUPERTYPE;
     assigned_security_classification : security_classification;
   END_ENTITY;

ENTITY organization_assignment
     ABSTRACT SUPERTYPE;
     assigned_organization : organization;
     role                  : organization_role;
   END_ENTITY;

ENTITY person_and_organization_assignment
     ABSTRACT SUPERTYPE;
     assigned_person_and_organization : person_and_organization;
     role                             : person_and_organization_role;
   END_ENTITY;

ENTITY date_and_time_assignment
     ABSTRACT SUPERTYPE;
     assigned_date_and_time : date_and_time;
     role                   : date_time_role;
   END_ENTITY;

ENTITY group_assignment
     ABSTRACT SUPERTYPE;
     assigned_group : group;
   END_ENTITY;

ENTITY document_type;
     product_data_type : label;
   END_ENTITY;

ENTITY document;
     id          : identifier;
     name        : label;
     description : text;
```

```
      kind        : document_type;
   UNIQUE
      UR1: id;
   END_ENTITY;

ENTITY document_usage_constraint;
      source                : document;
      subject_element       : label;
      subject_element_value : text;
   END_ENTITY;

ENTITY action;
      name          : label;
      description   : text;
      chosen_method : action_method;
   END_ENTITY;

ENTITY executed_action
      SUBTYPE OF (action);
   END_ENTITY;

ENTITY action_relationship;
      name            : label;
      description     : text;
      relating_action : action;
      related_action  : action;
   END_ENTITY;

ENTITY action_method;
      name        : label;
      description : text;
      consequence : text;
      purpose     : text;
   END_ENTITY;

ENTITY action_resource;
      name         : label;
      description  : text;
      usage        : supported_item;
      kind         : action_resource_type;
   END_ENTITY;

ENTITY action_resource_type;
      name : label;
   END_ENTITY;

ENTITY approval_status;
      name : label;
   END_ENTITY;

ENTITY approval;
      status : approval_status;
      level  : label;
   END_ENTITY;

ENTITY approval_date_time;
      date_time      : date_time_select;
      dated_approval : approval;
   END_ENTITY;

ENTITY approval_person_organization;
      person_organization : person_organization_select;
      authorized_approval : approval;
      role                : approval_role;
   END_ENTITY;

ENTITY approval_role;
      role : label;
```

```
   END_ENTITY;

ENTITY security_classification_level;
     name : label;
   END_ENTITY;

ENTITY security_classification;
     name           : label;
     purpose        : text;
     security_level : security_classification_level;
   END_ENTITY;

ENTITY address;
     internal_location       : OPTIONAL label;
     street_number           : OPTIONAL label;
     street                  : OPTIONAL label;
     postal_box              : OPTIONAL label;
     town                    : OPTIONAL label;
     region                  : OPTIONAL label;
     postal_code             : OPTIONAL label;
     country                 : OPTIONAL label;
     facsimile_number        : OPTIONAL label;
     telephone_number        : OPTIONAL label;
     electronic_mail_address : OPTIONAL label;
     telex_number            : OPTIONAL label;
   WHERE
     WR1: EXISTS(internal_location)       OR
          EXISTS(street_number)           OR
          EXISTS(street)                  OR
          EXISTS(postal_box)              OR
          EXISTS(town)                    OR
          EXISTS(region)                  OR
          EXISTS(postal_code)             OR
          EXISTS(country)                 OR
          EXISTS(facsimile_number)        OR
          EXISTS(telephone_number)        OR
          EXISTS(electronic_mail_address) OR
          EXISTS(telex_number);
   END_ENTITY;

ENTITY personal_address
     SUBTYPE OF (address);
     people      : SET [1:?] OF person;
     description : text;
   END_ENTITY;

ENTITY organizational_address
     SUBTYPE OF (address);
     organizations : SET [1:?] OF organization;
     description   : text;
   END_ENTITY;

ENTITY person;
     id            : identifier;
     last_name     : OPTIONAL label;
     first_name    : OPTIONAL label;
     middle_names  : OPTIONAL LIST [1:?] OF label;
     prefix_titles : OPTIONAL LIST [1:?] OF label;
     suffix_titles : OPTIONAL LIST [1:?] OF label;
   UNIQUE
     UR1: id;
   WHERE
     WR1: EXISTS(last_name) OR EXISTS(first_name);
   END_ENTITY;

ENTITY organization;
     id          : OPTIONAL identifier;
     name        : label;
```

```
     description : text;
   END_ENTITY;

ENTITY organizational_project;
     name                      : label;
     description               : text;
     responsible_organizations : SET[1:?] OF organization;
   END_ENTITY;

ENTITY person_and_organization;
     the_person       : person;
     the_organization : organization;
   END_ENTITY;

ENTITY person_and_organization_role;
     name : label;
   END_ENTITY;

ENTITY organization_role;
     name : label;
   END_ENTITY;

ENTITY date
     SUPERTYPE OF (ONEOF (calendar_date));
     year_component : year_number;
   END_ENTITY;

ENTITY calendar_date
     SUBTYPE OF (date);
     day_component   : day_in_month_number;
     month_component : month_in_year_number;
   WHERE
     WR1: valid_calendar_date (SELF);
   END_ENTITY;

ENTITY coordinated_universal_time_offset;
     hour_offset   : hour_in_day;
     minute_offset : OPTIONAL minute_in_hour;
     sense         : ahead_or_behind;
   END_ENTITY;

ENTITY local_time;
     hour_component   : hour_in_day;
     minute_component : OPTIONAL minute_in_hour;
     second_component : OPTIONAL second_in_minute;
     zone             : coordinated_universal_time_offset;
   WHERE
     WR1: valid_time (SELF);
   END_ENTITY;

ENTITY date_and_time;
     date_component : date;
     time_component : local_time;
   END_ENTITY;

ENTITY date_time_role;
     name : label;
   END_ENTITY;

ENTITY group;
     name        : label;
     description : text;
   END_ENTITY;

ENTITY effectivity;
     id    : identifier;
END_ENTITY;
```

```
ENTITY named_unit
     SUPERTYPE OF (ONEOF (si_unit, conversion_based_unit,
context_dependent_unit)
                    ANDOR
                    ONEOF (length_unit,
                           mass_unit,
                           time_unit,
                           electric_current_unit,
                           thermodynamic_temperature_unit,
                           amount_of_substance_unit,
                           luminous_intensity_unit,
                           plane_angle_unit,
                           solid_angle_unit,
                           area_unit,
                           volume_unit,
                           ratio_unit ));
     dimensions : dimensional_exponents;
   END_ENTITY;

ENTITY si_unit
     SUBTYPE OF (named_unit);
     prefix     : OPTIONAL si_prefix;
     name       : si_unit_name;
   DERIVE
     SELF\named_unit.dimensions : dimensional_exponents
                                := dimensions_for_si_unit (SELF.name);
   END_ENTITY;

ENTITY conversion_based_unit
     SUBTYPE OF (named_unit);
     name              : label;
     conversion_factor : measure_with_unit;
   END_ENTITY;

ENTITY context_dependent_unit
     SUBTYPE OF (named_unit);
     name : label;
   END_ENTITY;

ENTITY length_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent = 1.0)
    AND
          (SELF\named_unit.dimensions.mass_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.time_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.electric_current_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0)
AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
   END_ENTITY;

ENTITY mass_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent                    =
0.0) AND
          (SELF\named_unit.dimensions.mass_exponent                      =
1.0) AND
          (SELF\named_unit.dimensions.time_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.electric_current_exponent          =
0.0) AND
```

```
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent       =
0.0) AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent        =
0.0);
   END_ENTITY ;

ENTITY time_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent                    =
0.0) AND
          (SELF\named_unit.dimensions.mass_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.time_exponent                      =
1.0) AND
          (SELF\named_unit.dimensions.electric_current_exponent          =
0.0) AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent       =
0.0) AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent        =
0.0);
   END_ENTITY;

ENTITY electric_current_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent                    =
0.0) AND
          (SELF\named_unit.dimensions.mass_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.time_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.electric_current_exponent          =
1.0) AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent       =
0.0) AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent        =
0.0);
   END_ENTITY;

ENTITY thermodynamic_temperature_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent                    =
0.0) AND
          (SELF\named_unit.dimensions.mass_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.time_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.electric_current_exponent          =
0.0) AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
1.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent       =
0.0) AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent        =
0.0);
   END_ENTITY;

ENTITY amount_of_substance_unit
     SUBTYPE OF (named_unit);
   WHERE
```

```
     WR1: (SELF\named_unit.dimensions.length_exponent                    =
0.0) AND
              (SELF\named_unit.dimensions.mass_exponent
= 0.0) AND
              (SELF\named_unit.dimensions.time_exponent
= 0.0) AND
              (SELF\named_unit.dimensions.electric_current_exponent
= 0.0) AND
              (SELF\named_unit.dimensions.thermodynamic_temperature_exponent
= 0.0) AND
              (SELF\named_unit.dimensions.amount_of_substance_exponent
= 1.0) AND
              (SELF\named_unit.dimensions.luminous_intensity_exponent
= 0.0);
   END_ENTITY;

ENTITY luminous_intensity_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent                    =
0.0) AND
          (SELF\named_unit.dimensions.mass_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.time_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.electric_current_exponent          =
0.0) AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent       =
0.0) AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent        =
1.0);
   END_ENTITY;

ENTITY plane_angle_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent                    =
0.0) AND
          (SELF\named_unit.dimensions.mass_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.time_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.electric_current_exponent          =
0.0) AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent       =
0.0) AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent        =
0.0);
   END_ENTITY;

ENTITY solid_angle_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent                    =
0.0) AND
          (SELF\named_unit.dimensions.mass_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.time_exponent                      =
0.0) AND
          (SELF\named_unit.dimensions.electric_current_exponent          =
0.0) AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent       =
```

```
0.0) AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent        =
0.0);
   END_ENTITY;

ENTITY area_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent = 2.0)
    AND
          (SELF\named_unit.dimensions.mass_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.time_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.electric_current_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
   END_ENTITY;

ENTITY volume_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent = 3.0)
    AND
          (SELF\named_unit.dimensions.mass_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.time_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.electric_current_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
   END_ENTITY;

ENTITY ratio_unit
     SUBTYPE OF (named_unit);
   WHERE
     WR1: (SELF\named_unit.dimensions.length_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.mass_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.time_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.electric_current_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.thermodynamic_temperature_exponent =
0.0) AND
          (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0)
    AND
          (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
   END_ENTITY;

ENTITY dimensional_exponents;
     length_exponent                    : REAL;
     mass_exponent                      : REAL;
     time_exponent                      : REAL;
     electric_current_exponent          : REAL;
     thermodynamic_temperature_exponent : REAL;
     amount_of_substance_exponent       : REAL;
     luminous_intensity_exponent        : REAL;
   END_ENTITY;
```

```
     END_IF;

     IF 'TAS_SCHEMA.SOLID_ANGLE_MEASURE' IN TYPEOF ( m.value_component
) THEN
       IF derive_dimensional_exponents ( m.unit_component ) <>
          dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
         RETURN (FALSE);
       END_IF;
     END_IF;

     IF 'TAS_SCHEMA.AREA_MEASURE' IN TYPEOF ( m.value_component ) THEN
       IF derive_dimensional_exponents ( m.unit_component ) <>
          dimensional_exponents ( 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
         RETURN (FALSE);
       END_IF;
     END_IF;

     IF 'TAS_SCHEMA.VOLUME_MEASURE' IN TYPEOF ( m.value_component )
THEN
       IF derive_dimensional_exponents ( m.unit_component ) <>
          dimensional_exponents ( 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
         RETURN (FALSE);
       END_IF;
     END_IF;

     IF 'TAS_SCHEMA.RATIO_MEASURE' IN TYPEOF ( m.value_component ) THEN
       IF derive_dimensional_exponents ( m.unit_component ) <>
          dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
         RETURN (FALSE);
       END_IF;
     END_IF;

     IF 'TAS_SCHEMA.POSITIVE_LENGTH_MEASURE' IN TYPEOF (
m.value_component ) THEN
       IF derive_dimensional_exponents ( m.unit_component ) <>
          dimensional_exponents ( 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
         RETURN (FALSE);
       END_IF;
     END_IF;

     IF 'TAS_SCHEMA.POSITIVE_PLANE_ANGLE_MEASURE' IN TYPEOF (
m.value_component ) THEN
       IF derive_dimensional_exponents ( m.unit_component ) <>
          dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
         RETURN (FALSE);
       END_IF;
     END_IF;

     RETURN (TRUE);

   END_FUNCTION;

FUNCTION derive_dimensional_exponents (x : unit) :
dimensional_exponents;

     LOCAL
       i      : INTEGER;
       result : dimensional_exponents :=
                dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
     END_LOCAL;

     IF 'TAS_SCHEMA.DERIVED_UNIT' IN TYPEOF(x) THEN
       REPEAT i := LOINDEX(x.elements) TO HIINDEX(x.elements);

         result.length_exponent                    :=
           result.length_exponent +
           (x.elements[i].exponent *
            x.elements[i].unit.dimensions.length_exponent);
```

```
          result.mass_exponent                      :=
            result.mass_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.mass_exponent);

         result.time_exponent :=
            result.time_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.time_exponent);

          result.electric_current_exponent          :=
            result.electric_current_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.electric_current_exponent);

          result.thermodynamic_temperature_exponent :=
            result.thermodynamic_temperature_exponent +
            (x.elements[i].exponent *

x.elements[i].unit.dimensions.thermodynamic_temperature_exponent);

          result.amount_of_substance_exponent       :=
            result.amount_of_substance_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.amount_of_substance_exponent);

          result.luminous_intensity_exponent        :=
            result.luminous_intensity_exponent +
            (x.elements[i].exponent *
             x.elements[i].unit.dimensions.luminous_intensity_exponent);

        END_REPEAT;
      ELSE
        result := x.dimensions;
      END_IF;
      RETURN (result);
    END_FUNCTION;

FUNCTION dimensions_for_si_unit (n : si_unit_name) :
dimensional_exponents;

      CASE n  OF
        metre          : RETURN (dimensional_exponents
                                 (1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
        gram           : RETURN (dimensional_exponents
                                 (0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0));
        second         : RETURN (dimensional_exponents
                                 (0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0));
        ampere         : RETURN (dimensional_exponents
                                 (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0));
        kelvin         : RETURN (dimensional_exponents
                                 (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
        mole           : RETURN (dimensional_exponents
                                 (0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0));
        candela        : RETURN (dimensional_exponents
                                 (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
        radian         : RETURN (dimensional_exponents
                                 (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
        steradian      : RETURN (dimensional_exponents
                                 (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
        hertz          : RETURN (dimensional_exponents
                                 (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
        newton         : RETURN (dimensional_exponents
                                 (1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
        pascal         : RETURN (dimensional_exponents
                                 (-1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
        joule          : RETURN (dimensional_exponents
```

```
                                  (2.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
        watt           : RETURN (dimensional_exponents
                                  (2.0, 1.0, -3.0, 0.0, 0.0, 0.0, 0.0));
        coulomb        : RETURN (dimensional_exponents
                                  (0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0));
        volt           : RETURN (dimensional_exponents
                                  (2.0, 1.0, -3.0, -1.0, 0.0, 0.0, 0.0));
        farad          : RETURN (dimensional_exponents
                                  (-2.0, -1.0, 4.0, 1.0, 0.0, 0.0, 0.0));
        ohm            : RETURN (dimensional_exponents
                                  (2.0, 1.0, -3.0, -2.0, 0.0, 0.0, 0.0));
        siemens        : RETURN (dimensional_exponents
                                  (-2.0, -1.0, 3.0, 2.0, 0.0, 0.0, 0.0));
        weber          : RETURN (dimensional_exponents
                                  (2.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
        tesla          : RETURN (dimensional_exponents
                                  (0.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
        henry          : RETURN (dimensional_exponents
                                  (2.0, 1.0, -2.0, -2.0, 0.0, 0.0, 0.0));
        degree_Celsius : RETURN (dimensional_exponents
                                  (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
        lumen          : RETURN (dimensional_exponents
                                  (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
        lux            : RETURN (dimensional_exponents
                                  (-2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
        becquerel      : RETURN (dimensional_exponents
                                  (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
        gray           : RETURN (dimensional_exponents
                                  (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
        sievert        : RETURN (dimensional_exponents
                                  (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
      END_CASE;
    END_FUNCTION;

FUNCTION make_array_of_array(lis : LIST[1:?] OF LIST [1:?] OF GENERIC : T;
                              low1, u1, low2, u2 : INTEGER):
                  ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
    LOCAL
      n1,n2 : INTEGER;
      res   : ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
      resl  : LIST[1:?] OF ARRAY [low2:u2] OF GENERIC : T;
    END_LOCAL;


    n1 := SIZEOF(lis);
    n2 := SIZEOF(lis[1]);

    IF (n1 <> (u1 -low1 + 1)) AND (n2 <> (u2 - low2 + 1)) THEN
      RETURN(?);
    END_IF;

    REPEAT i := 1 TO n1;
      IF (SIZEOF(lis[i]) <> n2) THEN
        RETURN(?);
      END_IF;
    END_REPEAT;


    REPEAT i := 1 TO  n1;
      RESL[i] := list_to_array(lis[i],low2,u2);
    END_REPEAT;

    res := list_to_array(resl,low1,u1);
    RETURN(res);
  END_FUNCTION;

FUNCTION list_to_array(lis : LIST [0:?] OF GENERIC : T;
                       low,u : INTEGER) : ARRAY[low:u] OF GENERIC : T;
```

```
   LOCAL
     n   : INTEGER;
     res : ARRAY [low:u] OF GENERIC : T;
   END_LOCAL;

   n := SIZEOF(lis);
   IF (n <> (u-low +1)) THEN
     RETURN(?);
   ELSE
     REPEAT i := 1 TO n;
       res[low+i-1] := lis[i];
     END_REPEAT;
     RETURN(res);
   END_IF;
 END_FUNCTION;

FUNCTION surface_weights_positive(b: rational_b_spline_surface) : BOOLEAN;
   LOCAL
     result        : BOOLEAN := TRUE;
   END_LOCAL;

   REPEAT i := 0 TO b.u_upper;
     REPEAT j := 0 TO b.v_upper;
       IF (b.weights[i][j] <= 0.0)  THEN
         result := FALSE;
         RETURN(result);
       END_IF;
     END_REPEAT;
   END_REPEAT;
   RETURN(result);
 END_FUNCTION;

FUNCTION get_basis_surface (c : curve_on_surface) : SET[0:2] OF surface;
  LOCAL
    surfs  : SET[0:2] OF surface;
    n      : INTEGER;
  END_LOCAL;
  surfs := [];
  IF 'TAS_SCHEMA.PCURVE' IN TYPEOF (c) THEN
    surfs := [c\pcurve.basis_surface];
  ELSE
    IF 'TAS_SCHEMA.SURFACE_CURVE' IN TYPEOF (c) THEN
      n := SIZEOF(c\surface_curve.associated_geometry);
      REPEAT i := 1 TO n;
      surfs := surfs +
                 associated_surface(c\surface_curve.associated_geometry[i]);
      END_REPEAT;
    END_IF;
  END_IF;
  IF 'TAS_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF (c) THEN

     n := SIZEOF(c\composite_curve.segments);
     surfs := get_basis_surface(c\composite_curve.segments[1]);
     IF n > 1 THEN
       REPEAT i := 2 TO n;
         surfs := surfs * get_basis_surface(c\composite_curve.segments[i]);
       END_REPEAT;
     END_IF;
  END_IF;
  RETURN(surfs);
END_FUNCTION;

FUNCTION curve_weights_positive(b: rational_b_spline_curve) : BOOLEAN;
   LOCAL
     result : BOOLEAN := TRUE;
   END_LOCAL;

   REPEAT i := 0 TO b.upper_index_on_control_points;
```

```
      IF b.weights[i] <= 0.0  THEN
        result := FALSE;
        RETURN(result);
      END_IF;
    END_REPEAT;
    RETURN(result);
  END_FUNCTION;

FUNCTION constraints_param_b_spline(degree, up_knots, up_cp : INTEGER;
                                                    knot_mult : LIST OF INTEGER;
                        knots : LIST OF parameter_value) : BOOLEAN;
    LOCAL
      result  : BOOLEAN := TRUE;
      k,l,sum : INTEGER;
    END_LOCAL;


    sum := knot_mult[1];

    REPEAT i := 2 TO up_knots;
      sum := sum + knot_mult[i];
    END_REPEAT;


    IF (degree < 1) OR (up_knots < 2) OR (up_cp < degree) OR
          (sum <> (degree + up_cp + 2)) THEN
      result := FALSE;
      RETURN(result);
    END_IF;

    k := knot_mult[1];

    IF (k < 1) OR (k > degree + 1) THEN
      result := FALSE;
      RETURN(result);
    END_IF;

    REPEAT i := 2 TO up_knots;
      IF (knot_mult[i] < 1) OR (knots[i] <= knots[i-1]) THEN
        result := FALSE;
        RETURN(result);
      END_IF;

      k := knot_mult[i];

      IF (i < up_knots) AND (k > degree) THEN
        result := FALSE;
        RETURN(result);
      END_IF;

      IF (i = up_knots) AND (k > degree + 1) THEN
        result := FALSE;
        RETURN(result);
      END_IF;
    END_REPEAT;
    RETURN(result);
  END_FUNCTION;

FUNCTION vector_difference(arg1, arg2 : vector_or_direction) : vector;
    LOCAL
      result          : vector;
      res, vec1, vec2 : direction;
      mag, mag1, mag2 : REAL;
      ndim            : INTEGER;
    END_LOCAL;

    IF ((NOT EXISTS (arg1)) OR (NOT EXISTS (arg2))) OR (arg1.dim <>
arg2.dim)
```

```
        THEN
      result := ?;

    ELSE
      BEGIN
        IF 'TAS_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
          mag1 := arg1.magnitude;
          vec1 := arg1.orientation;
        ELSE
          mag1 := 1.0;
          vec1 := arg1;
        END_IF;
        IF 'TAS_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
          mag2 := arg2.magnitude;
          vec2 := arg2.orientation;
        ELSE
          mag2 := 1.0;
          vec2 := arg2;
        END_IF;
        vec1 := normalise (vec1);
        vec2 := normalise (vec2);
        ndim := SIZEOF(vec1.direction_ratios);
        mag := 0.0;
        REPEAT i := 1 TO ndim;
          res.direction_ratios[i] := mag1*vec1.direction_ratios[i] -
                                     mag2*vec2.direction_ratios[i];
          mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
        END_REPEAT;
        IF (mag > 0.0 ) THEN
          result.magnitude := SQRT(mag);
          result.orientation := res;
        ELSE
          result.magnitude := 0.0;
          result.orientation := vec1;
        END_IF;
      END;
    END_IF;
    RETURN (result);
  END_FUNCTION;

FUNCTION scalar_times_vector (scalar : REAL; vec : vector_or_direction)
                                     : vector;
    LOCAL
      v      : direction;
      mag    : REAL;
      result : vector;
    END_LOCAL;

    IF NOT EXISTS (scalar) OR NOT EXISTS (vec) THEN
      result := ?;

    ELSE
      IF 'TAS_SCHEMA.VECTOR' IN TYPEOF (vec) THEN
        v   := vec.orientation;
        mag := scalar * vec.magnitude;
      ELSE
        v   := vec;
        mag := scalar;
      END_IF;
      IF (mag < 0.0 ) THEN
        REPEAT i := 1 TO SIZEOF(v.direction_ratios);
          v.direction_ratios[i] := -v.direction_ratios[i];
        END_REPEAT;
        mag := -mag;
      END_IF;
      result.orientation := normalise(v);
      result.magnitude   := mag;
    END_IF;
```

```
   RETURN (result);
END_FUNCTION;

FUNCTION normalise (arg : vector_or_direction) : vector_or_direction;
   LOCAL
     ndim   : INTEGER;
     v      : direction;
     result : vector_or_direction;
     vec    : vector;
     mag    : REAL;
   END_LOCAL;

   IF NOT EXISTS (arg) THEN
     result := ?;

   ELSE
     ndim := arg.dim;
     IF 'TAS_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
       BEGIN
         vec := arg;
         v := arg.orientation;

         IF arg.magnitude = 0.0 THEN
           RETURN(?);
         ELSE
           vec.magnitude := 1.0;
         END_IF;
       END;
     ELSE
       v := arg;
     END_IF;
     mag := 0.0;
     REPEAT  i := 1 TO ndim;
       mag := mag + v.direction_ratios[i]*v.direction_ratios[i];
     END_REPEAT;
     IF mag > 0.0 THEN
       mag := SQRT(mag);
       REPEAT  i := 1 TO ndim;
         v.direction_ratios[i] := v.direction_ratios[i]/mag;
       END_REPEAT;
       IF 'TAS_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
         vec.orientation := v;
         result := vec;
       ELSE
         result := v;
       END_IF;
     ELSE
       RETURN(?);
     END_IF;
   END_IF;
   RETURN (result);
 END_FUNCTION;

FUNCTION dot_product(arg1, arg2 : direction) : REAL;
   LOCAL
     scalar : REAL;
     vec1, vec2: direction;
     ndim : INTEGER;
   END_LOCAL;

   IF NOT EXISTS (arg1) OR NOT EXISTS (arg2) THEN
     scalar := ?;

   ELSE
     IF (arg1.dim <> arg2.dim) THEN
       scalar := ?;

     ELSE
```

```
        BEGIN
          vec1   := normalise(arg1);
          vec2   := normalise(arg2);
          ndim   := arg1.dim;
          scalar := 0.0;
          REPEAT  i := 1 TO ndim;
            scalar := scalar +
                        vec1.direction_ratios[i]*vec2.direction_ratios[i];
          END_REPEAT;
        END;
      END_IF;
    END_IF;
    RETURN (scalar);
  END_FUNCTION;

FUNCTION cross_product (arg1, arg2 : direction) : vector;
    LOCAL
      mag    : REAL;
      res    : direction;
      v1,v2  : LIST[3:3] OF REAL;
      result : vector;
    END_LOCAL;

    IF ( NOT EXISTS (arg1) OR (arg1.dim = 2)) OR
       ( NOT EXISTS (arg2) OR (arg2.dim = 2)) THEN
      RETURN(?);
    ELSE
      BEGIN
        v1  := normalise(arg1).direction_ratios;
        v2  := normalise(arg2).direction_ratios;
        res.direction_ratios[1] := (v1[2]*v2[3] - v1[3]*v2[2]);
        res.direction_ratios[2] := (v1[3]*v2[1] - v1[1]*v2[3]);
        res.direction_ratios[3] := (v1[1]*v2[2] - v1[2]*v2[1]);
        mag := 0.0;
        REPEAT i := 1 TO 3;
          mag := mag + res.direction_ratios[i]*res.direction_ratios[i];
        END_REPEAT;
        IF (mag > 0.0) THEN
          result.orientation := res;
          result.magnitude := SQRT(mag);
        ELSE
          result.orientation := arg1;
          result.magnitude := 0.0;
        END_IF;
        RETURN(result);
      END;
    END_IF;
  END_FUNCTION;

FUNCTION second_proj_axis(z_axis, x_axis, arg: direction) : direction;
    LOCAL
      y_axis : vector;
      v      : direction;
      temp   : vector;
    END_LOCAL;

    IF NOT EXISTS(arg) THEN
      v := direction([0.0,1.0,0.0]);
    ELSE
      v := arg;
    END_IF;

    temp   := scalar_times_vector(dot_product(v, z_axis), z_axis);
    y_axis := vector_difference(v, temp);
    temp   := scalar_times_vector(dot_product(v, x_axis), x_axis);
    y_axis := vector_difference(y_axis, temp);
    y_axis := normalise(y_axis);
    RETURN(y_axis.orientation);
```

```
 END_FUNCTION;

FUNCTION first_proj_axis(z_axis, arg : direction) : direction;
   LOCAL
     x_axis : direction;
     v      : direction;
     z      : direction;
     x_vec  : vector;
   END_LOCAL;

   IF (NOT EXISTS(z_axis)) OR (NOT EXISTS(arg)) OR (arg.dim <> 3) THEN
     x_axis := ?;
   ELSE
     z_axis := normalise(z_axis);
     IF NOT EXISTS(arg) THEN
       IF (z_axis <> direction([1.0,0.0,0.0])) THEN
         v := direction([1.0,0.0,0.0]);
       ELSE
         v := direction([0.0,1.0,0.0]);
       END_IF;
     ELSE
       IF ((cross_product(arg,z).magnitude) = 0.0) THEN
         RETURN (?);
       ELSE
         v := normalise(arg);
       END_IF;
     END_IF;
     x_vec := scalar_times_vector(dot_product(v, z), z_axis);
     x_axis := vector_difference(v, x_vec).orientation;
     x_axis := normalise(x_axis);
   END_IF;
   RETURN(x_axis);
 END_FUNCTION;

FUNCTION orthogonal_complement(vec : direction) : direction;
   LOCAL
     result : direction;
   END_LOCAL;

   IF (vec.dim <> 2) OR NOT EXISTS (vec) THEN
     RETURN(?);
   ELSE
     result.direction_ratios[1] := -vec.direction_ratios[2];
     result.direction_ratios[2] := vec.direction_ratios[1];
     RETURN(result);
   END_IF;
 END_FUNCTION;

FUNCTION build_axes(axis, ref_direction : direction) :
                                          LIST [3:3] OF direction;
   LOCAL
     u : LIST[3:3] OF direction;
   END_LOCAL;

   u[3] := NVL(normalise(axis), direction([0.0,0.0,1.0]));
   u[1] := first_proj_axis(u[3],ref_direction);
   u[2] := normalise(cross_product(u[3],u[1])).orientation;
   RETURN(u);
 END_FUNCTION;

FUNCTION build_2axes(ref_direction : direction) : LIST [2:2] OF direction;
   LOCAL
     u : LIST[2:2] OF direction;
   END_LOCAL;

   u[1] := NVL(normalise(ref_direction), direction([1.0,0.0]));
   u[2] := orthogonal_complement(u[1]);
   RETURN(u);
```

```
 END_FUNCTION;

FUNCTION base_axis(dim : INTEGER; axis1, axis2, axis3 : direction) :
                                                   LIST [2:3] OF direction;
   LOCAL
     vec    : direction;
     u      : LIST [2:3] OF direction;
     factor : REAL;
   END_LOCAL;

   IF (dim = 3) THEN
     u[3] := NVL(normalise(axis3), direction([0.0,0.0,1.0]));
     u[1] := first_proj_axis(u[3],axis1);
     u[2] := second_proj_axis(u[3],u[1],axis2);
   ELSE
     u[3] := ?;
     IF EXISTS(axis1) THEN
       u[1] := normalise(axis1);
       u[2] := orthogonal_complement(u[1]);
       IF EXISTS(axis2) THEN
         factor := dot_product(axis2,u[2]);
         IF (factor < 0.0) THEN
           u[2].direction_ratios[1] := -u[2].direction_ratios[1];
           u[2].direction_ratios[2] := -u[2].direction_ratios[2];
         END_IF;
       END_IF;
     ELSE
       IF EXISTS(axis2) THEN
         u[2] := normalise(axis2);
         u[1] := orthogonal_complement(u[2]);
         u[1].direction_ratios[1] := -u[1].direction_ratios[1];
         u[1].direction_ratios[2] := - u[1].direction_ratios[2];
       ELSE
         u[1].direction_ratios[1] := 1.0;
         u[1].direction_ratios[2] := 0.0;
         u[2].direction_ratios[1] := 0.0;
         u[2].direction_ratios[2] := 1.0;
       END_IF;
     END_IF;
   END_IF;
   RETURN(u);
 END_FUNCTION;

FUNCTION associated_surface(arg : pcurve_or_surface) : surface;
   LOCAL
     surf : surface;
   END_LOCAL;

   IF 'TAS_SCHEMA.PCURVE' IN TYPEOF(arg) THEN
     surf := arg.basis_surface;
   ELSE
     surf := arg;
   END_IF;
   RETURN(surf);
 END_FUNCTION;

FUNCTION acyclic_surface_replica(rep : surface_replica; parent : surface)
                                                  : BOOLEAN;
   IF NOT (('TAS_SCHEMA.SURFACE_REPLICA') IN TYPEOF(parent)) THEN
      RETURN (TRUE);
   END_IF;

   IF (parent :=: rep) THEN
      RETURN (FALSE);

    ELSE RETURN(acyclic_surface_replica(rep,
                   parent\surface_replica.parent_surface));
    END_IF;
```

```
  END_FUNCTION;

FUNCTION acyclic_curve_replica(rep : curve_replica; parent : curve)
                                                  : BOOLEAN;
   IF NOT (('TAS_SCHEMA.CURVE_REPLICA') IN TYPEOF(parent)) THEN
      RETURN (TRUE);
   END_IF;

   IF (parent :=: rep) THEN
      RETURN (FALSE);

    ELSE RETURN(acyclic_curve_replica(rep,
parent\curve_replica.parent_curve));
    END_IF;
  END_FUNCTION;

FUNCTION dimension_of(item : geometric_representation_item) :
  dimension_count;
  LOCAL
    x : SET OF representation;
    y : representation_context;
  END_LOCAL;

  x := using_representations(item);

    y := x[1].context_of_items;
    RETURN (y\geometric_representation_context.coordinate_space_dimension);

END_FUNCTION;

FUNCTION list_to_set(l : LIST [0:?] OF GENERIC:T) : SET OF GENERIC:T;
   LOCAL
     s : SET OF GENERIC:T := [];
   END_LOCAL;

   REPEAT i := 1 TO SIZEOF(l);
     s := s + l[i];
   END_REPEAT;

   RETURN(s);
 END_FUNCTION;

FUNCTION mixed_loop_type_set(l: SET[0:?] OF loop): LOGICAL;
    LOCAL
      i             : INTEGER;
      poly_loop_type: LOGICAL;
    END_LOCAL;
    IF(SIZEOF(l) <= 1) THEN
      RETURN(FALSE);
    END_IF;
    poly_loop_type := ('TAS_SCHEMA.POLY_LOOP' IN TYPEOF(l[1]));
    REPEAT i := 2 TO SIZEOF(l);
      IF(('TAS_SCHEMA.POLY_LOOP' IN TYPEOF(l[i])) <> poly_loop_type)
THEN
         RETURN(TRUE);
       END_IF;
    END_REPEAT;
    RETURN(FALSE);
  END_FUNCTION;

FUNCTION list_face_loops(f: face) : LIST[0:?] OF loop;
   LOCAL
     loops : LIST[0:?] OF loop := [];
   END_LOCAL;

   REPEAT i := 1 TO SIZEOF(f.bounds);
     loops := loops +(f.bounds[i].bound);
   END_REPEAT;
```

```
   RETURN(loops);
 END_FUNCTION;

FUNCTION path_head_to_tail(a_path : path) : boolean;
   LOCAL
     n : INTEGER;

     p : LOGICAL := TRUE;
   END_LOCAL;

     n := SIZEOF (a_path.edge_list);
     REPEAT i := 2 TO n;
       p := p AND (a_path.edge_list[i-1].edge_end :=:
                   a_path.edge_list[i].edge_start);
     END_REPEAT;

     RETURN (p);
 END_FUNCTION;

FUNCTION boolean_choose (b : boolean;
           choice1, choice2 : GENERIC:item) : GENERIC:item;

   IF b THEN
     RETURN (choice1);
   ELSE
     RETURN (choice2);
   END_IF;
 END_FUNCTION;

FUNCTION list_of_topology_reversed (a_list
                                  : list_of_reversible_topology_item)
                                  : list_of_reversible_topology_item;
   LOCAL
     the_reverse : list_of_reversible_topology_item;
   END_LOCAL;

   the_reverse := [];
   REPEAT i := 1 TO SIZEOF (a_list);
     the_reverse := topology_reversed (a_list [i]) + the_reverse;
   END_REPEAT;

   RETURN (the_reverse);
 END_FUNCTION;

FUNCTION set_of_topology_reversed (a_set :
set_of_reversible_topology_item)
                                          :
set_of_reversible_topology_item;
   LOCAL
     the_reverse : set_of_reversible_topology_item;
   END_LOCAL;

   the_reverse := [];
   REPEAT i := 1 TO SIZEOF (a_set);
     the_reverse := the_reverse + topology_reversed (a_set [i]);
   END_REPEAT;

   RETURN (the_reverse);
 END_FUNCTION;

FUNCTION shell_reversed (a_shell : shell) : shell;
   LOCAL
     the_reverse : shell;
   END_LOCAL;

   IF ('TAS_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
     the_reverse := oriented_open_shell(
```

```
                        a_shell\oriented_open_shell.open_shell_element,
                          (NOT (a_shell\oriented_open_shell.orientation)));
   ELSE
     IF ('TAS_SCHEMA.OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
       the_reverse := oriented_open_shell (a_shell, FALSE);
     ELSE
       IF ('TAS_SCHEMA.ORIENTED_CLOSED_SHELL' IN TYPEOF (a_shell) )
THEN
         the_reverse := oriented_closed_shell(

a_shell\oriented_closed_shell.closed_shell_element,
                        NOT(a_shell\oriented_closed_shell.orientation));
       ELSE
         IF ('TAS_SCHEMA.CLOSED_SHELL' IN TYPEOF (a_shell) ) THEN
           the_reverse := oriented_closed_shell (a_shell, FALSE);
         ELSE
           the_reverse := ?;
         END_IF;
       END_IF;
     END_IF;
  END_IF;

  RETURN (the_reverse);
 END_FUNCTION;

FUNCTION face_reversed (a_face : face) : face;
   LOCAL
     the_reverse : face;
   END_LOCAL;

   IF ('TAS_SCHEMA.ORIENTED_FACE' IN TYPEOF (a_face) ) THEN
     the_reverse := oriented_face(a_face\oriented_face.face_element,
                                  (NOT
(a_face\oriented_face.orientation)));
   ELSE
     the_reverse := oriented_face (a_face, FALSE);
   END_IF;

   RETURN (the_reverse);
 END_FUNCTION;

FUNCTION face_bound_reversed (a_face_bound : face_bound) : face_bound;
   LOCAL
     the_reverse : face_bound;
   END_LOCAL;

   IF ('TAS_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF (a_face_bound) ) THEN
     the_reverse := a_face_bound;
     the_reverse\face_bound := face_bound(a_face_bound\face_bound.bound,
                                 (NOT(a_face_bound\face_bound.orientation)));
   ELSE
     the_reverse := face_bound(a_face_bound.bound,
                                (NOT (a_face_bound.orientation)));
   END_IF;

   RETURN (the_reverse);
 END_FUNCTION;

FUNCTION path_reversed (a_path : path) : path;
   LOCAL
     the_reverse : path;
   END_LOCAL;

   IF ('TAS_SCHEMA.ORIENTED_PATH' IN TYPEOF (a_path) ) THEN
     the_reverse := oriented_path(a_path\oriented_path.path_element,
                                   (NOT(a_path\oriented_path.orientation)));
   ELSE
     the_reverse := oriented_path (a_path, FALSE);
```

```
      END_IF;

      RETURN (the_reverse);
    END_FUNCTION;

   FUNCTION edge_reversed (an_edge : edge) : edge;
      LOCAL
        the_reverse : edge;
      END_LOCAL;

      IF ('TAS_SCHEMA.ORIENTED_EDGE' IN TYPEOF (an_edge) ) THEN
        the_reverse := oriented_edge(an_edge\oriented_edge.edge_element,
                                     (NOT (an_edge\oriented_edge.orientation)));
      ELSE
        the_reverse := oriented_edge (an_edge, FALSE);
      END_IF;
      RETURN (the_reverse);
    END_FUNCTION;

   FUNCTION topology_reversed (an_item : reversible_topology)
                                       : reversible_topology;

      IF ('TAS_SCHEMA.EDGE' IN TYPEOF (an_item)) THEN
        RETURN (edge_reversed (an_item));
      END_IF;

      IF ('TAS_SCHEMA.PATH' IN TYPEOF (an_item)) THEN
        RETURN (path_reversed (an_item));
      END_IF;

      IF ('TAS_SCHEMA.FACE_BOUND' IN TYPEOF (an_item)) THEN
        RETURN (face_bound_reversed (an_item));
      END_IF;

      IF ('TAS_SCHEMA.FACE' IN TYPEOF (an_item)) THEN
        RETURN (face_reversed (an_item));
      END_IF;

      IF ('TAS_SCHEMA.SHELL' IN TYPEOF (an_item)) THEN
        RETURN (shell_reversed (an_item));
      END_IF;

      IF ('SET' IN TYPEOF (an_item)) THEN
        RETURN (set_of_topology_reversed (an_item));
      END_IF;

      IF ('LIST' IN TYPEOF (an_item)) THEN
        RETURN (list_of_topology_reversed (an_item));
      END_IF;

      RETURN (?);
    END_FUNCTION;

   FUNCTION conditional_reverse (p       : BOOLEAN;
                                 an_item : reversible_topology)
                                         : reversible_topology;
      IF p THEN
        RETURN (an_item);
      ELSE
        RETURN (topology_reversed (an_item));
      END_IF;
    END_FUNCTION;

   FUNCTION constraints_geometry_shell_based_surface_model
                (m: shell_based_surface_model): BOOLEAN;
      LOCAL
        result : BOOLEAN := TRUE;
      END_LOCAL;
```

```
   REPEAT j := 1 TO SIZEOF(m.sbsm_boundary);
     IF (NOT ('TAS_SCHEMA.OPEN_SHELL' IN TYPEOF(m.sbsm_boundary[j]))
AND
        (NOT ('TAS_SCHEMA.CLOSED_SHELL' IN
TYPEOF(m.sbsm_boundary[j]))))
     THEN
       result := FALSE;
       RETURN(result);

     END_IF;
   END_REPEAT;

   RETURN(result);
 END_FUNCTION;

FUNCTION using_representations (item : representation_item)
  : SET OF representation;
  LOCAL
    results            : SET OF representation;
    result_bag         : BAG OF representation;
    intermediate_items : SET OF representation_item;
    i                  : INTEGER;
  END_LOCAL;

  result_bag := USEDIN(item,'TAS_SCHEMA.REPRESENTATION.ITEMS');

  IF SIZEOF(result_bag) > 0 THEN
    REPEAT i := 1 TO HIINDEX(result_bag);
      results := results + result_bag[i];
    END_REPEAT;
  END_IF;


  intermediate_items := QUERY(z <* bag_to_set( USEDIN(item , '')) |
    'TAS_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));

  IF SIZEOF(intermediate_items) > 0 THEN

    REPEAT i := 1 TO HIINDEX(intermediate_items);
      results := results + using_representations(intermediate_items[i]);
    END_REPEAT;
  END_IF;

  RETURN (results);
END_FUNCTION;

FUNCTION item_in_context
  (item  : representation_item;
   cntxt : representation_context) : BOOLEAN;
  LOCAL
    i : INTEGER;
    y : BAG OF representation_item;
  END_LOCAL;

  IF SIZEOF(USEDIN(item,'TAS_SCHEMA.REPRESENTATION.ITEMS')
    * cntxt.representations_in_context) > 0 THEN
    RETURN (TRUE);

    ELSE
    y := QUERY(z <* USEDIN (item , '') |
         'TAS_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));

      IF SIZEOF(y) > 0 THEN

      REPEAT i := 1 TO HIINDEX(y);

        IF item_in_context(y[i], cntxt) THEN
```

```
            RETURN (TRUE);
          END_IF;
        END_REPEAT;
      END_IF;
    END_IF;
    RETURN (FALSE);
  END_FUNCTION;

  FUNCTION acyclic_mapped_representation
    (parent_set   : SET OF representation;
     children_set : SET OF representation_item) : BOOLEAN;
    LOCAL
      x,y : SET OF representation_item;
      i   : INTEGER;
    END_LOCAL;

    x := QUERY(z <* children_set | 'TAS_SCHEMA.MAPPED_ITEM'
         IN TYPEOF(z));

    IF SIZEOF(x) > 0 THEN

      REPEAT i := 1 TO HIINDEX(x);

        IF x[i]\mapped_item.mapping_source.mapped_representation
          IN parent_set THEN
          RETURN (FALSE);
        END_IF;
        IF NOT acyclic_mapped_representation
          (parent_set +
  x[i]\mapped_item.mapping_source.mapped_representation,
          x[i]\mapped_item.mapping_source.mapped_representation.items) THEN
          RETURN (FALSE);
        END_IF;
      END_REPEAT;
    END_IF;

    x := children_set - x;

    IF SIZEOF(x) > 0 THEN

      REPEAT i := 1 TO HIINDEX(x);


        y := QUERY(z <* bag_to_set( USEDIN(x[i], '')) |
             'TAS_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));

        IF NOT acyclic_mapped_representation(parent_set, y) THEN
          RETURN (FALSE);
        END_IF;
      END_REPEAT;
    END_IF;
    RETURN (TRUE);
  END_FUNCTION;

  FUNCTION coordinated_pair_link_representation
      (link           : kinematic_link;
       pair_placement : rigid_placement) : BOOLEAN;
    LOCAL
      link_rep     : kinematic_link_representation;
    END_LOCAL;

    link_rep := representation_of_link (link);

    IF (link_rep = ?) THEN
      RETURN (FALSE);
    ELSE
      IF NOT (pair_placement IN link_rep\representation.items) THEN
        RETURN (FALSE);
```

```
    ELSE
      RETURN (TRUE);
    END_IF;
  END_IF;
END_FUNCTION;

FUNCTION unique_link_usage (link : kinematic_link) : BOOLEAN;
  LOCAL
    mechs  : SET OF mechanism;
    joints : SET OF kinematic_joint;
    struct : kinematic_structure;
  END_LOCAL;

  joints := bag_to_set
              (USEDIN (link,
                 'TAS_SCHEMA.KINEMATIC_JOINT.FIRST_LINK') +
               USEDIN (link,
                 'TAS_SCHEMA.KINEMATIC_JOINT.SECOND_LINK'));
  struct := joints[1].structure;

  REPEAT i := 2 TO SIZEOF (joints);
    IF (joints[i].structure :<>: struct) THEN
      RETURN (FALSE);
    END_IF;
  END_REPEAT;

  mechs := bag_to_set
             (USEDIN (struct,
                'TAS_SCHEMA.MECHANISM.STRUCTURE_DEFINITION'));

  IF (SIZEOF (mechs) <> 1) THEN
    RETURN (FALSE);
  END_IF;

  RETURN (TRUE);

END_FUNCTION;

FUNCTION representation_of_link (link : kinematic_link)
    : kinematic_link_representation;
  LOCAL
    link_rep_rel : BAG OF kinematic_link_representation_relation;
  END_LOCAL;

  link_rep_rel := USEDIN (link, 'TAS_SCHEMA.'+
                                'KINEMATIC_LINK_REPRESENTATION_RELATION.'+
                                'TOPOLOGICAL_ASPECTS');

  IF (SIZEOF (link_rep_rel) = 0) THEN
    RETURN (?);
  ELSE
    RETURN (link_rep_rel[1].geometric_aspects);
  END_IF;
END_FUNCTION;

FUNCTION acyclic (arg1 : generic_expression;
                      arg2 : SET OF generic_expression): BOOLEAN;
      LOCAL
            result: BOOLEAN;
      END_LOCAL;
      IF ('ISO_13584_GENERIC_EXPRESSIONS_SCHEMA.SIMPLE_GEN_EXPRESSION'
             IN TYPEOF (arg1))
      THEN RETURN (TRUE);
      END_IF;

      IF arg1 IN arg2
      THEN RETURN (FALSE);
      END_IF;
```

```
      IF 'ISO_13584_GENERIC_EXPRESSIONS_SCHEMA.UNARY_GEN_EXPRESSION'
            IN TYPEOF (arg1)
      THEN RETURN (acyclic(arg1\unary_gen_expression.operand,arg2+[arg1]));
      END_IF;

      IF 'ISO_13584_GENERIC_EXPRESSIONS_SCHEMA.BINARY_GEN_EXPRESSION'
            IN TYPEOF (arg1)
      THEN RETURN (acyclic(arg1\binary_gen_expression.operands[1],
                  arg2+[arg1]) AND acyclic(arg1\binary_gen_expression.operands[2],
                  arg2+[arg1]));
      END_IF;

      IF 'ISO_13584_GENERIC_EXPRESSIONS_SCHEMA.M_ARY_GEN_EXPRESSION'
            IN TYPEOF (arg1)
      THEN result := TRUE;
      REPEAT i := 1 TO SIZEOF (arg1\m_ary_gen_expression.operands);
            result := result AND
            acyclic(arg1\m_ary_gen_expression.operands[i], arg2+[arg1]);
      END_REPEAT;
      RETURN (result);
      END_IF;
END_FUNCTION;

FUNCTION is_acyclic (arg : generic_expression): BOOLEAN;
      RETURN (acyclic (arg, []));
END_FUNCTION ;

FUNCTION syntactic_variable_type (arg : variable): STRING;
IF 'TAS_SCHEMA.INT_NUMERIC_VARIABLE' IN TYPEOF (arg)
THEN
      RETURN ('INTEGER');
END_IF;
IF 'TAS_SCHEMA.REAL_NUMERIC_VARIABLE'IN TYPEOF (arg)
THEN
      RETURN ('REAL');
END_IF;
IF 'TAS_SCHEMA.BOOLEAN_VARIABLE' IN TYPEOF (arg)
THEN
      RETURN ('BOOLEAN');
END_IF;
IF 'TAS_SCHEMA_STRING_VARIABLE' IN TYPEOF (arg)
THEN
      RETURN ('STRING');
END_IF;
RETURN ('ERROR_TYPE');
END_FUNCTION;

FUNCTION syntactic_boolean_expression_type (arg : boolean_expression):
                  STRING;
LOCAL
      u_arg : unary_B_expression;
      b_arg : binary_B_expression ;
      m_arg : m_ary_B_expression ;
      c_arg : comparison_expression ;
      i_arg : interval_expression;
      i : INTEGER ;
END_LOCAL ;
IF 'TAS_SCHEMA.BOOLEAN_LITERAL' IN TYPEOF (arg)
THEN
      RETURN ('BOOLEAN');
END_IF;
IF 'TAS_SCHEMA.VARIABLE' IN TYPEOF (arg)
THEN
      RETURN (syntactic_variable_type (arg)) ;
END_IF;

IF 'TAS_SCHEMA.UNARY_B_EXPRESSION' IN TYPEOF (arg)
```

```
THEN
      u_arg := arg\unary_B_expression;

IF 'TAS_SCHEMA.NOT_EXPRESSION' IN TYPEOF (u_arg)
THEN
      IF syntactic_boolean_expression_type(u_arg.operand) ='BOOLEAN'
      THEN
            RETURN ('BOOLEAN');
      ELSE
            RETURN ('ERROR_TYPE');
      END_IF;
END_IF;

RETURN ('ERROR_TYPE');

END_IF ;

IF 'TAS_SCHEMA.BINARY_B_EXPRESSION' IN TYPEOF (arg)
THEN
      b_arg := arg\binary_B_expression;

RETURN ('ERROR_TYPE');
END_IF;

IF 'TAS_SCHEMA.M_ARY_B_EXPRESSION' IN TYPEOF (arg)
THEN
      m_arg := arg\m_ary_B_expression ;

IF 'TAS_SCHEMA.AND_EXPRESSION' IN TYPEOF (m_arg)
THEN
REPEAT i := 1 TO SIZEOF (m_arg.operands);
      IF (syntactic_boolean_expression_type(m_arg.operands[i]) <>
'BOOLEAN')
      THEN
            RETURN ('ERROR_TYPE');
      END_IF;
END_REPEAT;
RETURN ('BOOLEAN');
END_IF;

      RETURN ('ERROR_TYPE');
END_IF;

IF 'TAS_SCHEMA.COMPARISON_EXPRESSION' IN TYPEOF (arg)
THEN
      C_ARG := arg\comparison_expression ;

IF'TAS_SCHEMA.COMPARISON_GREATER_EQUAL'IN
TYPEOF(c_arg)
THEN
      IF (syntactic_expression_type (c_arg.operands [1]) = 'STRING')
                  AND (syntactic_expression_type (c_arg.operands [2]) = 'STRING')
      THEN
            RETURN ('BOOLEAN');
      END_IF;
      IF ((syntactic_expression_type (c_arg.operands[1]) = 'INTEGER')
                  OR (syntactic_expression_type (c_arg.operands [1]) = 'REAL'))
                  AND ((syntactic_expression_type (c_arg.operands [2]) = 'INTEGER')
                  OR (syntactic_expression_type (c_arg.operands [2]) = 'REAL'))
      THEN
            RETURN ('BOOLEAN');
      ELSE
            RETURN ('ERROR_TYPE');
      END_IF;
END_IF;

IF 'TAS_SCHEMA.COMPARISON_LESS_EQUAL' IN TYPEOF
(c_arg)
```

```
THEN
      IF (syntactic_expression_type (c_arg.operands [1]) = 'STRING')
                  AND (syntactic_expression_type (c_arg.operands [2]) = 'STRING')
      THEN
            RETURN ('BOOLEAN');
      END_IF;
      IF ((syntactic_expression_type (c_arg.operands[1]) = 'INTEGER')
                  OR (syntactic_expression_type (c_arg.operands [1]) = 'REAL'))
                  AND ((syntactic_expression_type (c_arg.operands [2]) = 'INTEGER')
                  OR (syntactic_expression_type (c_arg.operands [2]) = 'REAL'))
      THEN
            RETURN ('BOOLEAN');
      ELSE
            RETURN ('ERROR_TYPE');
      END_IF;
END_IF;

IF 'TAS_SCHEMA.INTERVAL_EXPRESSION' IN TYPEOF (c_arg)
THEN
      i_arg:=arg\interval_expression ;
      IF ((syntactic_expression_type(i_arg.interval_low)='STRING')
                  AND (syntactic_expression_type(i_arg.interval_item)='STRING')
                  AND (syntactic_expression_type(i_arg.interval_high)='STRING'))
      THEN
      RETURN ('BOOLEAN') ;
      END_IF;
      IF ((syntactic_expression_type(i_arg.interval_low)='BOOLEAN')
                  AND (syntactic_expression_type(i_arg.interval_item)='BOOLEAN')
                  AND (syntactic_expression_type(i_arg.interval_high)='BOOLEAN'))
      THEN
            RETURN ('BOOLEAN') ;
      END_IF;
      IF ((syntactic_expression_type(i_arg.interval_low)='INTEGER')
                  OR (syntactic_expression_type(i_arg.interval_low)='REAL'))
                  AND ((syntactic_expression_type(i_arg.interval_item)='INTEGER')
                  OR (syntactic_expression_type(i_arg.interval_item)='REAL'))
                  AND ((syntactic_expression_type(i_arg.interval_high)='INTEGER')
                  OR (syntactic_expression_type(i_arg.interval_high)='REAL'))
      THEN
            RETURN ('BOOLEAN');
      END_IF;
      RETURN ('ERROR_TYPE');
END_IF;

RETURN ('ERROR_TYPE');
END_IF;
END_FUNCTION;

FUNCTION syntactic_numeric_expression_type (arg : numeric_expression) : STRING;
LOCAL
      i : INTEGER ;
      u_arg : unary_n_expression ;
      m_arg : m_ary_n_expression ;
      b_arg : binary_n_expression ;
      c_arg : comparison_expression ;
      i_arg : interval_expression ;
      bool : BOOLEAN ;
END_LOCAL ;

IF 'TAS_SCHEMA.INT_LITERAL' IN TYPEOF (arg)
      THEN RETURN ('INTEGER');
END_IF;
IF 'TAS_SCHEMA.REAL_LITERAL' IN TYPEOF (arg)
      THEN RETURN ('REAL');
END_IF;
IF 'TAS_SCHEMA.VARIABLE' IN TYPEOF (arg)
      THEN RETURN (syntactic_variable_type (arg));
END_IF;
```

```
IF 'TAS_SCHEMA.UNARY_N_EXPRESSION' IN TYPEOF (arg)
THEN
      u_arg:= arg\unary_n_expression;

      RETURN ('ERROR_TYPE');
END_IF ;

IF 'TAS_SCHEMA.BINARY_N_EXPRESSION' IN TYPEOF (arg)
        THEN b_arg := arg\binary_n_expression;


      IF 'TAS_SCHEMA.POWER_EXPRESSION' IN TYPEOF (b_arg)
      THEN
            IF (syntactic_numeric_expression_type(b_arg.operands [1]) =
'INTEGER')
            THEN
                  IF (syntactic_numeric_expression_type(b_arg.operands[2]) =
'INTEGER')
                  THEN
                  RETURN ('INTEGER') ;
                  END_IF;
                  IF (syntactic_numeric_expression_type(b_arg.operands [2]) =
'REAL')
                  THEN
                  RETURN ('REAL');
                  ELSE
                  RETURN ('ERROR_TYPE');
                  END_IF;
            END_IF;
            IF (syntactic_numeric_expression_type(b_arg.operands [1]) = 'REAL')
            THEN
                  IF (syntactic_numeric_expression_type(b_arg.operands[2]) =
'INTEGER')
                  OR (syntactic_numeric_expression_type(b_arg.operands[2])=
'REAL')
                  THEN
                  RETURN ('REAL');
                  ELSE
                  RETURN ('ERROR_TYPE');
                  END_IF;
            END_IF;
      END_IF;
      RETURN ('ERROR_TYPE');

END_IF;

IF 'TAS_SCHEMA.M_ARY_N_EXPRESSION' IN TYPEOF (arg)
THEN
      m_arg := arg\m_ary_n_expression ;

IF 'TAS_SCHEMA.PLUS_EXPRESSION' IN TYPEOF (m_arg)
THEN bool := TRUE;
REPEAT i := 1 TO SIZEOF(m_arg.operands);
      IF (syntactic_numeric_expression_type(m_arg.operands[i])
                  <> 'INTEGER')
      THEN bool:= FALSE ;
      END_IF;
END_REPEAT;
      IF bool THEN RETURN ('INTEGER');
      END_IF;
      bool := TRUE;
      REPEAT i := 1 TO SIZEOF(m_arg.operands);
            IF (syntactic_numeric_expression_type(m_arg.operands[i])
                   <> 'REAL' )
                   AND (syntactic_numeric_expression_type(m_arg.operands[i])
                   <> 'INTEGER' )
            THEN bool:= FALSE ;
```

```
            END_IF;
      END_REPEAT;
      IF bool THEN RETURN ('REAL');
      ELSE RETURN ('ERROR_TYPE');
      END_IF;
END_IF;

IF 'TAS_SCHEMA.MULT_EXPRESSION' IN TYPEOF (m_arg)
THEN bool := TRUE;
REPEAT i := 1 TO SIZEOF(m_arg.operands);
      IF (syntactic_numeric_expression_type(m_arg.operands[i])
                  <> 'INTEGER')
      THEN bool:= FALSE ;
      END_IF;
END_REPEAT;
      IF bool THEN RETURN ('INTEGER');
      END_IF;
      bool := TRUE;
      REPEAT i := 1 TO SIZEOF(m_arg.operands);
            IF (syntactic_numeric_expression_type(m_arg.operands[i]) <> 'REAL'
)
                  AND (syntactic_numeric_expression_type(m_arg.operands[i])
                  <> 'INTEGER' )
            THEN bool:= FALSE ;
            END_IF;
      END_REPEAT;
      IF bool
      THEN
            RETURN ('REAL');
      ELSE
            RETURN ('ERROR_TYPE');
      END_IF;
END_IF;


RETURN ('ERROR_TYPE');
END_IF;

END_FUNCTION;

FUNCTION syntactic_expression_type (arg : expression) : STRING;
IF 'TAS_SCHEMA.NUMERIC_EXPRESSION' IN TYPEOF (arg)
THEN
      RETURN (syntactic_numeric_expression_type (arg)) ;
END_IF;
IF 'TAS_SCHEMA.BOOLEAN_EXPRESSION' IN TYPEOF (arg)
THEN
      RETURN (syntactic_boolean_expression_type (arg));
END_IF;
END_FUNCTION;

FUNCTION is_SQL_mappable (arg: expression) : BOOLEAN;
      LOCAL
            i: INTEGER;
      END_LOCAL;
      IF    'TAS_SCHEMA.SIMPLE_N_EXPRESSION'
                  IN TYPEOF (arg)
      THEN
            RETURN (TRUE);
      END_IF;
      IF    'TAS_SCHEMA.SQL_MAPPABLE_DEFINED_FUNCTION'
                  IN TYPEOF (arg)
      THEN
            RETURN (TRUE);
      END_IF;
      IF    'TAS_SCHEMA.MINUS_FUNCTION' IN TYPEOF(arg)
      THEN
            RETURN (is_SQL_mappable(arg\unary_n_expression.operand));
```

```
      END_IF;

      IF    ('TAS_SCHEMA.PLUS_EXPRESSION' IN TYPEOF(arg))
                  OR ('TAS_SCHEMA.MULT_EXPRESSION' IN
TYPEOF(arg))
      THEN
            REPEAT i :=1 TO SIZEOF (arg\m_ary_n_expression.operands);
                  IF NOT is_SQL_mappable(arg\m_ary_n_expression.operands[i])
                  THEN
                  RETURN (FALSE);
                  END_IF;
            END_REPEAT;
            RETURN (TRUE);
      END_IF;
      IF 'TAS_SCHEMA.SIMPLE_B_EXPRESSION' IN TYPEOF (arg)
      THEN
            RETURN (TRUE);
      END_IF;
      IF ('TAS_SCHEMA.COMPARISON_GREATER_EQUAL'
                  IN TYPEOF (arg))
            OR ('TAS_SCHEMA.COMPARISON_LESS_EQUAL'
                  IN TYPEOF (arg))
      THEN
            RETURN (is_SQL_mappable (arg\COMPARISON_EXPRESSION.OPERANDS[1])
                  AND is_SQL_mappable (arg\COMPARISON_EXPRESSION.OPERANDS[2]));
      END_IF;

      IF 'TAS_SCHEMA.INTERVAL_EXPRESSION' IN TYPEOF(arg)
      THEN
            RETURN (is_SQL_mappable(arg\interval_expression.interval_low) AND
                      is_SQL_mappable(arg\interval_expression.interval_high) AND
                        is_SQL_mappable(arg\interval_expression.interval_item));
      END_IF;

      RETURN (FALSE);
END_FUNCTION;

FUNCTION is_int_expr (arg: numeric_expression) : BOOLEAN;
      LOCAL
            i: INTEGER;
      END_LOCAL;
      IF 'TAS_SCHEMA.INT_LITERAL' IN TYPEOF(arg)
      THEN
            RETURN (TRUE);
      END_IF;
      IF 'TAS_SCHEMA.REAL_LITERAL' IN TYPEOF(arg)
      THEN
            RETURN (FALSE);
      END_IF;
      IF 'TAS_SCHEMA.INT_NUMERIC_VARIABLE' IN TYPEOF(arg)
      THEN
            RETURN (TRUE);
      END_IF;
      IF 'TAS_SCHEMA.REAL_NUMERIC_VARIABLE' IN
TYPEOF(arg)
      THEN
            RETURN (FALSE);
      END_IF;

      IF ('TAS_SCHEMA.PLUS_EXPRESSION' IN TYPEOF(arg))
OR ('TAS_SCHEMA.MULT_EXPRESSION' IN TYPEOF(arg))
      THEN
            REPEAT i :=1 TO SIZEOF (arg\m_ary_n_expression.operands);
                  IF NOT is_int_expr(arg\m_ary_n_expression.operands[i])
                  THEN
                  RETURN (FALSE);
                  END_IF;
            END_REPEAT;
```

```
            RETURN (TRUE);
      END_IF;
      IF    ('TAS_SCHEMA.MINUS_EXPRESSION' IN TYPEOF(arg))
OR ('TAS_SCHEMA.POWER_EXPRESSION' IN TYPEOF(arg))
      THEN
            RETURN (is_int_expr(arg\binary_n_expression.operands[1])
                  AND is_int_expr(arg\binary_n_expression.operands[2]));
      END_IF;

      RETURN (FALSE);
END_FUNCTION;

FUNCTION consistent_size(parameters: LIST[1:?] OF list_ascending_real_literals;
function_values: list_real_literals):BOOLEAN;

LOCAL

      result: BOOLEAN;

      n_values: INTEGER;

      i: INTEGER;

END_LOCAL;

n_values := SIZEOF(parameters[1]);

REPEAT i:=2 TO SIZEOF(parameters);

      n_values := n_values * SIZEOF(parameters[i]);

END_REPEAT;

result := (SIZEOF(function_values) = n_values);

RETURN(result);

END_FUNCTION;

FUNCTION valid_bounded_variables(bounds:SET OF less_or_greater; variables: LIST OF
SIR_variable): BOOLEAN;

LOCAL

      i: INTEGER;

      bounded: SET OF BOOLEAN;

END_LOCAL;

IF (SIZEOF(bounds)>=SIZEOF(variables)) THEN RETURN(FALSE);

END_IF;

REPEAT i:=LOINDEX(variables) TO HIINDEX(variables);

      bounded[i]:=FALSE;

END_REPEAT;

REPEAT i:=LOINDEX(bounds) TO HIINDEX(bounds);

      IF NOT ('TAS_SCHEMA.LITERAL_NUMBER' IN
TYPEOF(bounds[i]\binary_gen_expression.operands[1])) THEN RETURN(FALSE);

      END_IF;

      IF NOT ('TAS_SCHEMA.SIR_VARIABLE' IN
```

```
TYPEOF(bounds[i]\binary_gen_expression.operands[2])) THEN RETURN(FALSE);

      END_IF;

END_REPEAT;

RETURN(TRUE);

END_FUNCTION;

FUNCTION ascending_list(values: list_real_literals): BOOLEAN;

LOCAL

      i: INTEGER;

END_LOCAL;

REPEAT i:= LOINDEX(values) TO (HIINDEX(values)-1);

      IF (values[i]>values[i+1]) THEN RETURN(FALSE);

      END_IF;

END_REPEAT;

RETURN(TRUE);

END_FUNCTION;

FUNCTION get_containers(prop: SIR_descriptive_or_functional): SET OF
SIR_list_of_values_or_functions;

LOCAL

      i: INTEGER;

      sav: BAG OF SIR_applicability_of_values;

      result: SET OF SIR_list_of_values_or_functions;

END_LOCAL;

sav := USEDIN( prop,
'TAS_SCHEMA.SIR_APPLICABILITY_OF_VALUES.VALID_USAGE.PROPERTY');

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);

      result[i]:=sav[i].valid_data;

END_REPEAT;

RETURN(result);

END_FUNCTION;

FUNCTION default_pointing : SIR_pointing_in_space;

LOCAL

      result: SIR_pointing_in_space;

END_LOCAL;

result\item_defined_transformation.transform_item_1 := direction([1.,0.,0.]);

result\item_defined_transformation.transform_item_2 :=
descriptive_representation_item('space coordinate system');
```

```
RETURN(result);
END_FUNCTION;
FUNCTION valid_velocity( mwu: measure_with_unit; numer: STRING): BOOLEAN;
LOCAL
      the_unit: unit;
END_LOCAL;
the_unit:= mwu.unit_component;
IF NOT('TAS_SCHEMA.DERIVED_UNIT' IN TYPEOF (the_unit)) THEN
      RETURN(FALSE);
END_IF;
IF (SIZEOF(the_unit.elements) <>2) THEN
      RETURN(FALSE);
END_IF;
IF NOT(('TAS_SCHEMA.' + numer)IN TYPEOF (the_unit.elements[1].unit)) THEN
      RETURN(FALSE);
END_IF;
IF NOT(('TAS_SCHEMA.' + numer)IN TYPEOF (the_unit.elements[2].unit)) THEN
      RETURN(FALSE);
END_IF;
IF NOT ((the_unit.elements[1].exponent = 1) AND
     (the_unit.elements[2].exponent = -1)) THEN
      RETURN(FALSE);
END_IF;
RETURN(TRUE);
END_FUNCTION;
FUNCTION strictly_ascending_list(values: LIST OF REAL): BOOLEAN;
LOCAL
      i: INTEGER;
END_LOCAL;
REPEAT i:= LOINDEX(values) TO (HIINDEX(values)-1);
      IF (values[i]>=values[i+1]) THEN RETURN(FALSE);
      END_IF;
END_REPEAT;
RETURN(TRUE);
```

```
END_FUNCTION;

FUNCTION basis_surface_check (surf : surface) : BOOLEAN;

  IF 'TAS_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(surf) THEN
    RETURN(TRUE);
  ELSE

    IF 'TAS_SCHEMA.SWEPT_SURFACE' IN TYPEOF (surf) THEN
      RETURN (basis_curve_check (surf\swept_surface.swept_curve));
    ELSE

      IF (('TAS_SCHEMA.OFFSET_SURFACE' IN TYPEOF (surf)) AND
        (surf\offset_surface.self_intersect = FALSE)) THEN
        RETURN (basis_surface_check (surf\offset_surface.basis_surface));
      ELSE

        IF 'TAS_SCHEMA.SURFACE REPLICA' IN TYPEOF(surf) THEN
          RETURN(basis_surface_check (surf\surface_replica.parent_surface));
        ELSE

          IF (('TAS_SCHEMA.B_SPLINE_SURFACE' IN TYPEOF(surf)) AND
            (surf\b_spline_surface.self_intersect = FALSE)) THEN
            RETURN(TRUE);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;

FUNCTION basis_curve_check (cv: curve) : BOOLEAN;

  IF (('TAS_SCHEMA.B_SPLINE_CURVE' IN TYPEOF (cv)) AND
    (cv\b_spline_curve.self_intersect = FALSE))
  THEN RETURN(TRUE);
  ELSE

    IF SIZEOF (['TAS_SCHEMA.BOUNDED_CURVE', 'TAS_SCHEMA.CONIC',
      'TAS_SCHEMA.LINE'] * TYPEOF (cv)) = 1 THEN
      RETURN(TRUE);
    ELSE

      IF SIZEOF (['TAS_SCHEMA.BOUNDED_CURVE',
        'TAS_SCHEMA.CURVE_REPLICA'] * TYPEOF (cv)) = 1 THEN
        RETURN (basis_curve_check(cv\curve_replica.parent_curve));
      ELSE

        IF ((SIZEOF (['TAS_SCHEMA.BOUNDED_CURVE',
          'TAS_SCHEMA.OFFSET_CURVE_3D'] * TYPEOF (cv)) = 1) AND
          (cv\offset_curve_3d.self_intersect = FALSE)) THEN
          RETURN (basis_curve_check (cv\offset_curve_3d.basis_curve));
        ELSE

          IF SIZEOF (['TAS_SCHEMA.BOUNDED_CURVE',
            'TAS_SCHEMA.PCURVE'] * TYPEOF(cv)) = 1 THEN
            RETURN ((basis_curve_check
            (cv\pcurve.reference_to_curve\representation.items[1])) AND
            (basis_surface_check (cv\pcurve.basis_surface)));

          ELSE

            IF SIZEOF (['TAS_SCHEMA.BOUNDED_CURVE',
              'TAS_SCHEMA.SURFACE_CURVE'] * TYPEOF(cv)) = 1 THEN

              IF basis_curve_check (cv\surface_curve.curve_3d) THEN
```

```
                REPEAT i := 1 TO SIZEOF
                (cv\surface_curve.associated_geometry);

                  IF 'TAS_SCHEMA.SURFACE' IN
                    TYPEOF (cv\surface_curve.associated_geometry[i]) THEN
                    IF NOT basis_surface_check
                      (cv\surface_curve.associated_geometry[i]) THEN
                      RETURN(FALSE);
                    END_IF;
                  ELSE
                    IF 'TAS_SCHEMA.PCURVE' IN
                      TYPEOF (cv\surface_curve.associated_geometry[i]) THEN
                      IF NOT basis_curve_check
                        (cv\surface_curve.associated_geometry[i]) THEN
                        RETURN(FALSE);
                      END_IF;
                    END_IF;
                  END_IF;
                END_REPEAT;
                RETURN(TRUE);
              END_IF;
            ELSE

              IF 'TAS_SCHEMA.POLYLINE' IN TYPEOF(cv) THEN
                IF (SIZEOF (cv\polyline.points) >= 3) THEN RETURN (TRUE);
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;


FUNCTION get_names(prop: named_item): SET OF label;
LOCAL
      i: INTEGER;
      sav: BAG OF TAS_name_assignment;
      result: SET OF label;
END_LOCAL;

sav := USEDIN( prop, 'TAS_SCHEMA.TAS_NAME_ASSIGNMENT.ITEMS');

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
      result[i]:=sav[i].assigned_name;
END_REPEAT;

RETURN(result);
END_FUNCTION;
(*
*)
FUNCTION get_tools(prop: NRF_run): SET OF action_resource;
LOCAL
      i: INTEGER;
      sav: BAG OF action_resource;
      result: SET OF action_resource;
END_LOCAL;

sav := USEDIN( prop, 'TAS_SCHEMA.ACTION_RESOURCE.USAGE');

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
      IF (sav[i].kind.name='tool_or_facility') THEN
      result[i]:=sav[i];
      END_IF;
END_REPEAT;
```

```
RETURN(result);
END_FUNCTION;
(*
*)
FUNCTION get_timestamps(prop: NRF_run): SET OF date_and_time;
LOCAL
      i: INTEGER;
      sav: BAG OF TAS_date_assignment;
      result: SET OF date_and_time;
END_LOCAL;

sav := USEDIN( prop, 'TAS_SCHEMA.TAS_DATE_ASSIGNMENT.ITEMS');

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
      IF (sav[i].role.name='start_date_and_time') THEN
      result[i]:=sav[i].assigned_date_and_time;
      END_IF;
END_REPEAT;

RETURN(result);
END_FUNCTION;
(*
*)
FUNCTION unique_ids(sav: BAG OF GENERIC): LOGICAL;
LOCAL
      i: INTEGER;
      bag_id: BAG OF identifier;
END_LOCAL;

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
      bag_id[i] := sav[i].id;
END_REPEAT;

RETURN(VALUE_UNIQUE(bag_id));
END_FUNCTION;
(*
*)
FUNCTION unique_names(sav: BAG OF GENERIC): LOGICAL;
LOCAL
      i: INTEGER;
      bag_names: BAG OF label;
END_LOCAL;

REPEAT i:=LOINDEX(sav) TO HIINDEX(sav);
      bag_names[i]:= sav[i].name;
END_REPEAT;

RETURN(VALUE_UNIQUE(bag_names));
END_FUNCTION;
(*
*)
FUNCTION valid_properties_of_celestial_body(item:TAS_celestial_body):BOOLEAN;
LOCAL
      result: BOOLEAN;
      prod_def_form_set: BAG OF product_definition_formation;
      prod_def_set : BAG OF product_definition;
      prop_def_set: BAG OF property_definition;
      i,j,k: INTEGER;
      nb_prop1, nb_prop2: INTEGER;
END_LOCAL;

prod_def_form_set := QUERY(pdf <* USEDIN(item,
'TAS_SCHEMA.PRODUCT_DEFINITION_FORMATION.OF_PRODUCT') |
('TAS_SCHEMA.SIR_PRODUCT_VERSION' IN TYPEOF(pdf)));

REPEAT k:=LOINDEX(prod_def_form_set) TO HIINDEX(prod_def_form_set);
```

```
      prod_def_set:= USEDIN(prod_def_form_set[k],
'TAS_SCHEMA.PRODUCTION_DEFINITION.FORMATION');
      nb_prop1:=0;
      nb_prop2:=0;

(* checks if the expected properties are present *)
      REPEAT i:=LOINDEX(prod_def_set) TO HIINDEX(prod_def_set);
         prop_def_set:= USEDIN(prod_def_set[i],
'TAS_SCHEMA.PROPERTY_DEFINITION.DEFINITION');
         REPEAT j:=LOINDEX(prop_def_set) TO HIINDEX(prop_def_set);
            IF ('TAS_SCHEMA.TAS_GRAVITATIONAL_ASPECTS_OF_CELESTIAL_BODY' IN
TYPEOF(prop_def_set[j])) THEN
               nb_prop1:= nb_prop1 + 1;
            END_IF;
            IF ('TAS_SCHEMA.TAS_THERMAL_ASPECTS_OF_CELESTIAL_BODY' IN
TYPEOF(prop_def_set[j])) THEN
               nb_prop2:= nb_prop2 + 1;
            END_IF;
         END_REPEAT;
   END_REPEAT;

      IF NOT ((nb_prop1=1) AND (nb_prop2<=1)) THEN
         RETURN(FALSE);
      END_IF;
END_REPEAT;

RETURN(TRUE);
END_FUNCTION;
(*
*)
FUNCTION orbit_assigned_to(item: TAS_celestial_body_with_orbit):BOOLEAN;
LOCAL
      prod_def_form_set: BAG OF product_definition_formation;
      prod_def_set: BAG OF product_definition;
      prop_def_set: BAG OF property_definition;
      orbit_set: BAG OF TAS_orbit_assignment;
      i,j,k: INTEGER;
      nb_orbit: INTEGER;
END_LOCAL;

prod_def_form_set := QUERY(pdf <* USEDIN(item,
'TAS_SCHEMA.PRODUCT_DEFINITION_FORMATION.OF_PRODUCT')|
('TAS_SCHEMA.SIR_PRODUCT_VERSION' IN TYPEOF(pdf)));

REPEAT i:=LOINDEX(prod_def_form_set) TO HIINDEX(prod_def_form_set);
      prod_def_set:= USEDIN(prod_def_form_set[i],
'TAS_SCHEMA.PRODUCTION_DEFINITION.FORMATION');
      nb_orbit:=0;

      REPEAT j:=LOINDEX(prod_def_set) TO HIINDEX(prod_def_set);
         prop_def_set:= USEDIN(prod_def_set[j],
'TAS_SCHEMA.PROPERTY_DEFINITION.DEFINITION');
         REPEAT k:=LOINDEX(prop_def_set) TO HIINDEX(prop_def_set);
            orbit_set:= USEDIN(prop_def_set[k],
'TAS_SCHEMA.TAS_ORBIT_ASSIGNMENT.RELATED_PROPERTY_DEFINITION');
           nb_orbit:= nb_orbit + SIZEOF(orbit_set);
        END_REPEAT;
      END_REPEAT;

      IF (nb_orbit<>1) THEN
         RETURN(FALSE);
      END_IF;
END_REPEAT;

RETURN(TRUE);
END_FUNCTION;
(*
*)
```

```
FUNCTION TAS_valid_associated_kepler_set(item:TAS_discretised_orbit_arc):BOOLEAN;
LOCAL
result: BOOLEAN;
auxr: SET OF representation_relationship;
END_LOCAL;

auxr:= USEDIN(item.geom_rep, 'TAS_SCHEMA.REPRESENTATION_RELATIONSHIP.REP_1');
result := (SIZEOF(auxr) <= 1);

IF result AND (SIZEOF(auxr) = 1) THEN
      result:= (auxr[1].description = 'ASSOCIATED UNPERTURBED ORBIT DATA') AND
('TAS_SCHEMA.TAS_SPACE_COORDINATE_SYSTEM' IN TYPEOF
(auxr[1].rep_2.context_of_items)) AND
(SIZEOF(auxr[1].rep_2.items) = 1);
IF result THEN
      result:= ('TAS_SCHEMA.TAS_KEPLER_SET' IN TYPEOF(auxr[1].rep_2.items[1]));
END_IF;
END_IF;

RETURN(result);
END_FUNCTION;
(*
*)
FUNCTION TAS_valid_albedo_reflector(item:TAS_albedo_reflector):BOOLEAN;
LOCAL
related: product_definition;
relating: product_definition;
result: BOOLEAN;
auxp, auxp2: BAG OF property_definition;
auxr: BAG OF property_definition_representation;
END_LOCAL;

related:= item\product_definition_relationship.related_product_definition;
relating:= item\product_definition_relationship.relating_product_definition;

result:= 'TAS_SCHEMA.TAS_CELESTIAL_BODY' IN TYPEOF(relating.formation.of_product);

IF (result) THEN
      result := (related :<>: relating);
END_IF;

IF (result) THEN
      auxp:= USEDIN(item, 'TAS_SCHEMA.PROPERTY_DEFINITION.DEFINITION');
      auxp2:= QUERY( prop_def <* auxp | (prop_def.name = 'albedo reflection
characteristics'));
      result := (SIZEOF(auxp2) = 1);
END_IF;

IF (result) THEN
      auxr:= QUERY(prdr<* USEDIN(auxp2,
'TAS_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') |
('TAS_SCHEMA.TAS_ALBEDO_REFLECTION_DATA' IN TYPEOF(prdr)));
      result:= (SIZEOF(auxr) =1);
END_IF;

RETURN(result);
END_FUNCTION;
(*
*)


END_SCHEMA;
```

Annex B - AIM Short Names for Entities (Normative)

This annex provides a unique assigned acronym for each entity name occuring in this protocol. The long form entity names have been extracted from the EXPRESS language statements found in the AIM EXPRESS listing of annex A.

Requirements on the use of the short names are found in the implementation methods included in ISO 10303.

As far as the AIM entities are extracted from existing integrated resources, the short name specified in ISO 10303 documents is used. Otherwise a new name has been developed.

Note: some entities have in this edition no associated short name; they shall be transferred in the neutral files with their long name.

Long Name	*Short Name*
action	ACTION
action_assignment	ACTASS
action_method	ACTMTH
action_relationship	ACTRLT
action_resource	ACTRSR
action_resource_type	ACRSTY
address	ADDRSS
advanced_face	ADVFC
amount_of_substance_measure_with_unit	AOSMWU
amount_of_substance_unit	AOSU
application_context	APPCNT
application_context_element	APCNEL
application_protocol_definition	APPRDF
approval	APPRVL
approval_assignment	APPASS
approval_date_time	APDTTM
approval_person_organization	APPROR
approval_role	APPRL
approval_status	APPSTT
area_measure_with_unit	AMWU
area_unit	ARUNT
assembly_component_usage	ASCMUS
axis1_placement	AX1PLC
axis2_placement_2d	A2PL2D
axis2_placement_3d	A2PL3D
bezier_curve	BZRCRV
bezier_surface	BZRSRF
bounded_curve	BNDCRV
bounded_pcurve	BNDPCR
bounded_surface	BNDSRF
bounded_surface_curve	BNSRCR
b_spline_curve	BSPCR

Long Name	*Short Name*
b_spline_curve_with_knots	BSCWK
b_spline_surface	BSPSR
b_spline_surface_with_knots	BSSWK
calendar_date	CLNDT
cartesian_point	CRTPNT
cartesian_transformation_operator	CRTROP
cartesian_transformation_operator_3d	CTO3
characterized_object	CHROBJ
circle	CIRCLE
closed_shell	CLSSHL
colour	COLOUR
colour_rgb	CLRRGB
colour_specification	CLRSPC
composite_curve	CPMCRV
composite_curve_segment	CMCRSG
conic	CONIC
conical_surface	CNCSRF
connected_face_set	CNFCST
context_dependent_unit	CNDPUN
conversion_based_unit	CNBSUN
coordinated_universal_time_offset	CUTO
curve	CURVE
curve_replica	CRVRPL
cylindrical_surface	CYLSRF
data_environment	DTENV
date	DATE
date_and_time	DTANTM
date_and_time_assignment	DATA
date_time_role	DTTMRL
definitional_representation	DFNRPR
degenerate_pcurve	DGNPCR
degenerate_toroidal_surface	DGTRSR
derived_unit	DRVUNT
derived_unit_element	DRVNEL
descriptive_representation_item	DSRPIT
dimensional_exponents	DMNEXP
direction	DRCTN
document	DCMNT
document_reference	DCMRFR
document_type	DCMTYP
document_usage_constraint	CNBSUN
edge	EDGE
edge_curve	EDGCRV
edge_loop	EDGLP
effectivity	EFFCTV
electric_current_measure_with_unit	ECMWU
electric_current_unit	ELCRUN

Long Name	*Short Name*
elementary_surface	ELMSRF
ellipse	ELLPS
evaluated_degenerate_pcurve	EVDGPC
executed_action	EXCACT
face	FACE
face_bound	FCBND
face_outer_bound	FCOTBN
face_surface	FCSRF
functionally_defined_transformation	FNDFTR
geometric_representation_context	GMRPCN
geometric_representation_item	GMRPIT
global_uncertainty_assigned_context	GC
global_unit_assigned_context	GUAC
group	GROUP
group_assignment	GRPASS
half_space_solid	HLSPSL
hyperbola	HYPRBL
intersection_curve	INTCRV
item_defined_transformation	ITDFTR
kinematic_ground_representation	KNGRRP
kinematic_joint	KNMLNT
kinematic_link	KNMLNK
kinematic_link_representation	KNLNRP
kinematic_link_representation_association	KLRA
kinematic_link_representation_relation	KLRR
kinematic_pair	KNMPR
kinematic_property_definition	KNPRDF
kinematic_property_representation_relation	KPRR
kinematic_structure	KNMSTR
length_measure_with_unit	LMWU
length_unit	LNGUNT
line	LINE
local_time	LCLTM
loop	LOOP
luminous_intensity_measure_with_unit	LIMWU
luminous_intensity_unit	LMINUN
manifold_surface_shape_representation	MSSR
mapped_item	MPPITM
mass_measure_with_unit	MMWU
mass_unit	MSSUNT
material_designation	MTRDSG
material_designation_characterization	MTDSCH
material_property	MTRPRP
material_property_representation	MTPRRP
measure_representation_item	MSRPIT
measure_with_unit	MSWTUN
mechanism	MCHNSM

Long Name	*Short Name*
name_assignment	NMASS
named_unit	NMDUNT
next_assembly_usage_occurrence	NAUO
NRF_derivation	NRFD
NRF_derivation_bounds	NRFDB
NRF_derivation_procedure	NRFDP
NRF_derivation_result	NRFDR
NRF_run	NRFR
NRF_run_in_sequence	NRFRIS
NRF_run_phase	NRFRP
NRF_run_results	NRFRR
NRF_scan_derived	NRFSD
NRF_scan_sampled	NRFSS
NRF_sequence	NRFS
NRF_sequence_case	NRFSC
offset_curve_3d	OFCR3D
offset_surface	OFFSRF
open_shell	OPNSHL
organization	ORGNZT
organizational_address	ORGADD
organization_assignment	ORGASS
organizational_project	ORGPRJ
organization_role	ORGRL
oriented_edge	ORNEDG
oriented_face	ORNFC
oriented_path	ORNPTH
pair_value	PRVL
parabola	PRBL
parametric_representation_context	PRRPCN
path	PATH
pcurve	PCURVE
person	PERSON
person_and_organization	PRANOR
person_and_organization_assignment	PAOA
person_and_organization_role	PAOR
personal_address	PRSADD
placement	PLCMNT
plane	PLANE
plane_angle_measure_with_unit	PAMWU
plane_angle_unit	PLANUN
point	POINT
point_on_curve	PNONCR
point_on_surface	PNONSR
polyline	PLYLN
prismatic_pair	PRSPR
prismatic_pair_range	PRPRRN
prismatic_pair_value	PRPRVL

Long Name	*Short Name*
product	PRDCT
product_category	PRDCTG
product_category_relationship	PRCTRL
product_context	PRDCNT
product_definition	PRDDFN
product_definition_context	PRDFCN
product_definition_formation	PRDFFR
product_definition_relationship	PRDFRL
product_definition_shape	PRDFSH
product_definition_usage	PRDFUS
product_related_product_category	PRPC
property_definition	PRPDFN
property_definition_representation	PRDFRP
quasi_uniform_curve	QSUNCR
quasi_uniform_surface	QSUNSR
ratio_measure_with_unit	RMWU
ratio_unit	RTUNT
rational_b_spline_curve	RBSC
rational_b_spline_surface	RBSS
representation	RPRSNT
representation_context	RPRCNT
representation_item	RPRITM
representation_map	RPRMP
representation_relationship	RPRRLT
revolute_pair	RVLPR
revolute_pair_range	RVPRRN
revolute_pair_value	RVPRVL
rotation_about_direction	RTABDR
seam_curve	SMCRV
security_classification	SCRCLS
security_classification_assignment	SCCLAS
security_classification_level	SCCLLV
si_unit	SUNT
simple_pair_range	SMPRRN
shape_aspect	SHPASP
shape_representation	SHPRPR
shell_based_surface_model	SBSM
SIR_applicability_of_values	SIRAOV
SIR_aspect	SIRA
SIR_ato_campaign	SIRCAM
SIR_ato_case	SIRCAS
SIR_ato_phase	SIRP
SIR_boolean_result_replica	SIRBRR
SIR_boolean_result_surface	SIRBRS
SIR_case_in_campaign	SIRCIC
SIR_colour_assignment	SIRCAS
SIR_component_sequence	SIRCS

Long Name	*Short Name*
SIR_cyclic_tabular_function	SIRCTF
SIR_event_assignment	SIREA
SIR_event_effectivity	SIREE
SIR_function_valued_property	SIRFVP
SIR_implicit_revolute_pair_value	SIRIRP
SIR_index_interval	SIRII
SIR_initial_conditioning	SIRIC
SIR_kepler_set	SIRKS
SIR_kinematic_articulation	SIRKA
SIR_kinematic_link	SIRKL
SIR_list_of_descriptive_values	SIRLDV
SIR_list_of_functions	SIRLOF
SIR_mask	SIRMA
SIR_model	SIRMO
SIR_node	SIRN
SIR_node_relationship	SIRNR
SIR_node_relationship_usage	SIRNRU
SIR_node_usage	SIRNU
SIR_number_valued_property	SIRNVP
SIR_parameterized_function	SIRPAF
SIR_placement_transformation	SIRPT
SIR_polynomial_function	SIRPOF
SIR_pointing_in_space	SIRPIS
SIR_position_and_velocity	SIRPAV
SIR_product_version	SIRPV
SIR_property_class	SIRPC
SIR_property_descriptive	SIRPD
SIR_property_functional	SIRPF
SIR_property_meaning	SIRPM
SIR_property_name	SIRPN
SIR_property_quantitative	SIRPQ
SIR_property_scalar	SIRPS
SIR_property_tensor	SIRPT
SIR_property_usage	SIRPU
SIR_rotation_motion	SIRRM
SIR_scalar_in_tensor	SIRSIT
SIR_scan	SIRS
SIR_space_aspect_in_case	SIRSAP
SIR_space_direction	SIRSD
SIR_space_mission_aspect	SIRSMA
SIR_space_mission_case	SIRSMC
SIR_space_mission_event	SIRSME
SIR_space_mission_phase	SIRSMP
SIR_string_valued_property	SIRSVP
SIR_submodel_usage	SIRSU
SIR_tabular_function	SIRTF
SIR_thermal_radiative_explicit_meshing	SIRTRE

Long Name	*Short Name*
SIR_thermal_radiative_face_set	SIRTRF
SIR_thermal_radiative_incremental_meshing	SIRTRI
SIR_unit_assignment	SIRUA
SIR_unit_with_offset	SIRUWO
SIR_variable	SIRV
solid_angle_measure_with_unit	SAMWU
solid_angle_unit	SLANUN
spherical_surface	SPHSRF
surface	SRFC
surface_curve	SRFCRV
surface_of_linear_extrusion	SL
surface_of_revolution	SROFRV
surface_replica	SRFRPL
surface_side_style	SRSDST
surface_style_boundary	SRSTBN
surface_style_usage	SRSTUS
swept_surface	SWPSRF
TAS_albedo_reflector	TASARE
TAS_albedo_reflection_data	TASARD
TAS_approval_assignment	TASAA
TAS_basic_thermal_radiative_model	TASBTM
TAS_body_motion_representation	TASBMR
TAS_celestial_body	TASCB
TAS_celestial_body_with_orbit	TASCBO
TAS_colour_assignment	TASCA
TAS_contact_assignment	TASCNA
TAS_commercial_designation	TASCD
TAS_date_assignment	TASDA
TAS_discretised_orbit_arc	TASDOR
TAS_document_reference	TASDOC
TAS_enclosure	TASEN
TAS_evaluation_event	TASEE
TAS_gravitational_aspects_of_celestial_body	TASGAB
TAS_gravitational_constant	TASGC
TAS_kepler_orbit_arc	TASKOR
TAS_material_property	TASMP
TAS_material_property_environment	TASMPE
TAS_meshed_thermal_radiative_model	TASMTM
TAS_model_product_relationship	TASMPR
TAS_name_assignment	TASNA
TAS_notional_distance	TASND
TAS_orbit_arc	TASORB
TAS_orbit_assignment	TASOA
TAS_orbit_governor	TASOG
TAS_revolution_period	TASRP
TAS_security_assignment	TASSA
TAS_thermal_aspects_of_celestial_body	TASTAB

Long Name	*Short Name*
TAS_thermal_environment	TASTE
TAS_thermal_environment_data	TASTED
TAS_thermal_radiation_spectrum	TASRSP
TAS_space_coordinate_system	TASSCS
TAS_thermal_flux	TASTFM
thermodynamic_temperature_measure_with_unit	TTMWU
thermodynamic_temperature_unit	THTMUN
time_measure_with_unit	TMWU
time_unit	TMUNT
topological_representation_item	TPRPIT
toroidal_surface	TRDSRF
translation	TRNSLT
type_qualifier	TYPQLF
uncertainty_measure_with_unit	UMWU
uniform_curve	UNFCRV
uniform_surface	UNFSRF
vector	VECTOR
vertex	VERTEX
vertex_loop	VRTLP
vertex_point	VRTPNT
volume_measure_with_unit	VMWU
volume_unit	VLMUNT
binary_gen_expression	ISO13584_BIGEEX
binary_n_expression	ISO13584_BINEX
boolean_expression	ISO13584_BOOEXP
comparison_expression	ISO13584_COMEXP
comparison_greater_equal	ISO13584_COGREQ
comparison_less_equal	ISO13584_CLOE
environment	ISO13584_ENVIRO
expression	ISO13584_EXPRES
generic_expression	ISO13584_GENEXP
generic_literal	ISO13584_GENLIT
generic_variable	ISO13584_GENVAR
int_literal	ISO13584_INTLIT
int_numeric_variable	ISO13584_INNUVA
literal_number	ISO13584_LITNUM
mathematical_string	ISO13584_MATSTR
mult_expression	ISO13584_MULEXP
m_ary_gen_expression	ISO13584_MAGE
m_ary_n_expression	ISO13584_MANE
numeric_expression	ISO13584_NUMEXP
numeric_variable	ISO13584_NUMVAR
plus_expression	ISO13584_PLUEXP
power_expression	ISO13584_POWEXP
real_literal	ISO13584_REALIT
real_numeric_variable	ISO13584_RENUVA
simple_gen_expresssion	ISO13584_SIGEEX

Long Name	*Short Name*
unary_gen_expression	ISO13584_UNGEEX
unary_n_expression	ISO13584_UNNEX
simple_n_expression	ISO13584_SINEX
variable	ISO13584_VARIAB
variable_semantics	ISO13584_VARSEM

Annex C - Implementation Method Specific Requirements (Normative)

An implementation method defines the exchange characteristics and representation that are required by this part. Conformance to this part shall be realized using one of the implementation methods specified in ISO 10303.

This annex specifies additional requirements for the use of this part of ISO 10303 with an exchange structure as specified in ISO 10303-21.

C.1 Reference to the AIM schema

This annex specifies requirements on the header section of an exchange structure.

The attribute **schema_identifiers** of the entity **file_name** of the header section of an exchange structure shall for the purpose of this part include in its list of **schema_names** the name '**TAS_SCHEMA**' and only this name.

C.2 Instance of the entity application_protocol_definition

An instance of the entity **application_protocol_definition** shall be coded in each neutral file. The values of its first three attributes shall be:

- **status**: 'STEP-TAS VERSION 2 - RELEASE 2.4'
- **application_interpreted_model_schema_name**: 'TAS_SCHEMA'
- **application_protocol_year**: 1997

C.3 Normative names for classes and roles of valued properties

In this protocol the values defined below shall be used for the names and roles of valued properties – i.e. for instances of the entities **SIR_property_name** and **SIR_property_meaning** – whenever they are applicable for the property that is being instanciated.

C.3.1 Values for SIR_property_name

For the **name** attribute of **SIR_property_name** the following values are defined:

value	*definition*	*preferred SI unit*	*remarks*
'absorptance'	A thermo-optical quantity denoting the ratio of absorbed over incident radiation.	-	
'acceleration'	The time rate of change of velocity.	$m.s^{-2}$	
'amount_of_substance'	The number of elementary entities of which a substance consists.	mol	The elementary entities can be atoms, molecules, ions, electrons, other particles, or specfied groups of such particles.
'angle'	Identical to 'plane_angle'.	rad	This shall only be used via **SIR_scan** instances.

value	*definition*	*preferred SI unit*	*remarks*
'cartesian_position'	Position in three Cartesian coordinates.	m	The three components are denoted by roles 'x', 'y' and 'z'. This shall only be used via **SIR_scan** instances.
'cartesian_velocity'	Velocity in three Cartesian components.	$m.s^{-1}$	The three components are denoted by roles 'x', 'y' and 'z'. This shall only be used via **SIR_scan** instances.
'charge'	Electric charge.	C	
'count'	Count of a number of discrete items.	-	
'density'	Mass per unit volume.	$kg.m^{-3}$	
'electric_current'	TBD – see ISO 31 and ISO 1000.	A	
'electric_conductance'	TBD – see ISO 31 and ISO 1000.	siemens	
'electric_potential'	TBD – see ISO 31 and ISO 1000.	V	
'electric_resistance'	TBD – see ISO 31 and ISO 1000.	ohm	
'emittance'	A thermo-optical quantity denoting the ratio of the amount of emitted radiation of a specific body over the amount emitted by a perfect black body at identical temperature.	-	
'energy'			
'flux'	Power per unit area.	$W.m^{-2}$	
'force'	TBD – see ISO 31 and ISO 1000.	N	
'frequency'	TBD – see ISO 31 and ISO 1000.	Hz	
'heat_capacity'	TBD – see ISO 31 and ISO 1000.	$J.kg^{-1}.K^{-1}$	
'irradiance'	TBD – see ISO 31 and ISO 1000.	$W.m^{-2}$	
'length'	TBD – see ISO 31 and ISO 1000.	m	This shall only be used via **SIR_scan** instances.
'luminous_intensity'	TBD – see ISO 31 and ISO 1000.	cd	
'mass'	TBD – see ISO 31 and ISO 1000.	kg	
'moment_of_force'	TBD – see ISO 31 and ISO 1000.	N.m	
'moment_of_inertia'	TBD – see ISO 31 and ISO 1000.	$kg.m^2$	
'plane_angle'	TBD – see ISO 31 and ISO 1000.	rad	
'polar_position'	Position in polar coordinates.	m, rad	The three components are denoted by roles 'radius', 'phi' and 'z'.
'power'	Energy per unit time.	W	
'pressure'	Force per unit area.	Pa	

value	*definition*	*preferred SI unit*	*remarks*
'ratio'	TBD – see ISO 31 and ISO 1000.	-	The range of a ratio is not constrained to be in [0,1] interval, except if this is explicitely stated in the definition of the property_ meaning.
'reflectance'	A thermo-optical quantity denoting the ratio of reflected over incident radiation.	-	
'relative_humidity'	TBD – see ISO 31 and ISO 1000.	-	
'solid_angle'	TBD – see ISO 31 and ISO 1000.	sr	
'spherical_position'	Position in spherical coordinates	m, rad	The three components are denoted by roles 'radius', 'phi' and 'theta'.
'strain'	TBD – see ISO 31 and ISO 1000.	-	
'stress'	TBD, as in ISO 31 and ISO 1000.	$N.m^{-2}$	
'temperature'	Identical to 'thermodynamic_ temperature'.	K, °C	
'thermal_conductivity'	A material property, denoting the linear thermal conductance for a rectangular parallellopipid of unit length and cross sectional area.	$W.m^{-1}.K^{-1}$	
'thermal_conductance'	The linear thermal conductance between two reference points in a specific physical object or medium.	$W.K^{-1}$	
'thermal_radiative_ exchange_factor'	The ratio of thermal radiative exchange between two thermal-radiative faces i and j, which is the fraction of the thermal radiation emitted by face i that is finally – possibly after multiple reflections – absorbed by face j.	-	
'thermal_radiative_ coupling'	Denotes the radiative coupling between two thermal-radiative faces i and j, which is the product of the surface area of face i and the radiative exchange factor from i to j and the emittance of face i.	m^2	The coupling can alternatively be defined as the product of the surface area of face j and the radiative exchange factor from j to i and the emittance of face j.
'thermodynamic_ temperature'	TBD – see ISO 31 and ISO 1000.	K, °C	
'time'	TBD – see ISO 31 and ISO 1000.	s	
'transmittance'	A thermo-optical quantity denoting the ratio of transmitted over incident radiation.	-	
'velocity'	TBD – see ISO 31 and ISO 1000.	$m.s^{-1}$	

value	*definition*	*preferred SI unit*	*remarks*
'viewfactor'	The ratio for the direct view from one geometric face to another. The viewfactor from face i to face j is defined as the quotient of the solid angle representing the direct view from i to j over the solid angle of total possible view from face i.	-	For a flat face the total possible view is a solid angle of 2π steradians, i.e. a hemisphere.
'voltage'	TBD – see ISO 31 and ISO 1000.	V	
'volume'	TBD – see ISO 31 and ISO 1000.	m^3	
'wavelength'	TBD – see ISO 31 and ISO 1000.	nm	
'work'	TBD – see ISO 31 and ISO 1000.	J	

C.3.2 Values for roles in SIR_property_meaning

For the **type_qualifier** list elements of the **roles** attribute in **SIR_property_meaning** the following values are defined:

value	*definition*	*remarks*
'absolute'	Qualified as absolute-valued within some frame of reference.	Mutually exclusive with 'relative'.
'absorbed'	Due to absorption of radiation.	E.g. 'absorbed' 'solar' 'power'.
'alarm'	Qualification of an alarm situation.	E.g. 'upper' 'alarm' 'temperature'.
'albedo'	Qualification of the thermo-optical property 'reflectance'. The albedo reflectance specifies the reflection factor for the reflection of thermal radiation from a star (typically Sun) at a planet's surface and/or atmosphere.	When no further qualification is given, 'albedo' 'reflectance' is considered to be diffuse.
'ambient'	Of the ambient environment.	E.g. 'ambient' 'temperature'.
'axial'	Along the axial direction in a cylindrically symmetric coordinate system.	E.g. 'axial' 'velocity'. Mutually exclusive with 'radial'.
'diffuse'	Qualification of a thermo-optical property (i.e. 'absorptance', 'reflectance', 'transmittance' or 'emittance'), that specifies that the outgoing direction of thermal radiation is indepedent from the incident direction and evenly distributed over a hemi-sphere.	E.g. 'diffuse' 'solar' 'absorptance'.
'displacement'	Qualification of a 'length', which denotes a movement from a nominal position.	

value	*definition*	*remarks*
'distance'	Qualification of a 'length', which denotes a 'length' from a starting position.	
'dynamic'	Varying with time.	Mutually exclusive with 'static'.
'external'	On the outside.	Mutually exclusive with 'internal'.
'final'	At the end of a time interval.	Mutually exclusive with 'initial'.
'hemi_spherical'	Over the whole hemi-sphere.	Mutually exclusive with 'normal'. E.g. 'hemi_spherical' 'diffuse' 'emittance'.
'incident'	Pertaining to the incoming radiation.	Qualification of power or flux, e.g. 'incident' 'infrared' 'power'.
'infra_red'	Pertaining to the infra-red thermal radiation part of the electro-magnetic radation spectrum.	Mutually exclusive with 'solar' and 'ultra_violet'.
'initial'	At the beginning of a time interval.	Mutually exclusive with 'final'.
'internal'	On the inside.	Mutually exclusive with 'external'.
'latitude'	Qualification of a 'plane_angle' to denote a position on a celestial body. Its range is –90° ≤ latitude angle ≤ +90°. The positive direction is defined as northward.	0° latitude is the equatorial plane. +90° is the North Pole. -90° is the South Pole.
'limit'	Qualification of the extreme of an interval.	
'linear'	Exhibiting or assumed to exhibit linear behaviour.	Mutually exclusive with 'non_linear'.
'longitude'	Qualification of a 'plane_angle' to denote a position on a celestial body. Its range is –180° ≤ longitude angle ≤ +180°.	The positive direction is defined as eastward from the 0° longitude, which is also called the 0° meridian.
'lower'	At the lower end of some interval or range.	Mutually exclusive with 'upper'.
'maximum'	Pertaining to the maximum value over a population of property values.	Mutually exclusive with 'minimum'.
'mean'	Pertaining to the mean value over a population of property values.	E.g. 'mean' 'ambient' 'pressure'.
'measured'	Qualification that a property value was measured.	Mutually exclusive with 'predicted' and 'computed'.
'minimum'	Pertaining to the minimum value over a population of property values.	Mutually exclusive with 'maximum'.
'negative'	Smaller than zero.	Mutually exclusive with 'positive'.
'nominal'	In the normal, prevailing situtation.	
'non_linear'	Exhibiting or assumed to exhibit non-linear behaviour.	Mutually exclusive with 'linear'.

value	*definition*	*remarks*
'normal'	With a direction making locally a 90° angle with a line or surface.	Mutually exclusive with 'hemi_ spherical'.
'phi'	Qualification of the ascension angle for polar and spherical coordinate systems.	
'planet'	Pertaining to a planet, including the thermal radiation that is emitted by a planet.	E.g. 'incident' 'planet' 'flux'. Mutually exclusive with 'solar' and 'star'.
'positive'	Greater than zero.	Mutually exclusive with 'negative'.
'predicted'	As derived in a prediction.	Mutually exclusive with 'measured'.
'radial'	Along the radial direction in a cylindrically symmetric coordinate system.	E.g. 'radial' 'velocity'. Mutually exclusive with 'axial'.
'radius'	Qualification of a 'length' from a centerpoint or centerline to a circle's circumference, a cylindircal surface or a spherical surface.	
'reflected'	Pertaining to reflected radiation.	
'relative'	Relative to some other proerty value in a certain context.	Mutually exclusive with 'absolute'.
'solar'	Pertaining to the Sun, including the thermal radiation that is emitted by the Sun.	E.g. 'solar' 'power' or 'solar' 'flux'. Mutually exclusive with 'infra_red' and 'planet'.
'static'	Constant over time.	Mutually exclusive with 'dynamic'.
'switch_on'	Pertaining to a property value for which switch on is considered safe.	Mutually exclusive with 'switch_off'. 'switch_on' 'temperature'.
'switch_off'	Pertaining to a property value for which switch off is considered necessary.	Mutually exclusive with 'switch_on'.
'theta'	Qualification of thedeclination angle in a spherical coordinate system.	Qualification of the elevation angle for spherical coordinate systems.
'total'	All taken together.	
'upper'	At the upper end of some interval or range.	Mutually exclusive with 'lower'.
'ultra_violet'	Qualification of part of the thermal radiative properties.	
'x'	Qualification of the x-component in a Cartesian coordinate system.	Mutually exclusive with 'y' and 'z'.
'y'	Qualification of the y-component in a Cartesian coordinate system.	Mutually exclusive with 'x' and 'z'.
'z'	Qualification of the z-component in a Cartesian coordinate system.	Mutually exclusive with 'x' and 'y'.

Annex D - Protocol Implementation Conformance Statement (PICS) proforma (Normative)

This clause lists the optional elements of this application protocol. An implementation may chose to support any combinations of the optional elements. However, certain combinations of options are likely to be implemented together. These combinations are called conformance classes and are described in the subclauses of this annex.

This annex is in the form of a questionnaire. This questionnaire is intended to be filled out by the implementor and may be used in preparation for conformance testing by a testing laboratory. The completed PICS proforma is referred to as a PICS.

WORKING NOTE 1 – PICS proforma form to be added.

Annex E - TBD (Normative)

Annex F - Application Activity Model (Informative)

F.1 Application Activity Model Definitions and Abbreviations

The AAMs of [STEP-203] “Configuration Controlled Design” and [STEP-214] “Core Data for Automotive Mechanical Design Processes” were used as examples for the definition of this AAM. Where possible the same terminology of processes, inputs, controls, outputs and mechanisms was used.

The following terms are used in the AAM.

NOTE: Terms marked with an asterisk (*) are outside the scope of this Application Protocol.

F.1.1 **accepted product*** – product which is delivered for deployment and operation after successful acceptance testing.

F.1.2 **acceptance test specification*** – a test specification containing the requirements that must be fulfilled, the test setup and the test script for the acceptance test. Acceptance testing is done after manufacture, assembly and integration of a product are completed. After successful acceptance testing there is an operational product ready for delivery for deployment and operation can take place.

F.1.3 **analysis case** – definition of a case to be analysed, comprising the definition and assignment of all relevant analysis parameters.

F.1.4 **analysis model** – a representation – in software – of (part of) the product under development for a particular type of analysis.

F.1.5 **analysis results** – the results obtained from an analysis run with a particular analysis model for a particular analysis case.

F.1.6 **applicable standards*** – formalised documentation on parts, practices, procedures and formats that is applicable to the product development, production and/or operation activities. The objective of making standards applicable is to ensure certain quality or performance levels and to ensure commonality between products.

F.1.7 **breadboard facilities** – facilities for breadboard testing, i.e. concept or feasibility checks with relatively simple test setups.

F.1.8 **budget constraints*** – constraints imposing the budget for the product development.

F.1.9 **CAx tools** – Computer Aided Design, Engineering and/or Manufacturing tools.

F.1.10 **Conceptualise Product Design*** – the process of taking design ideas, requirements and design constraints, and turning them into a formalised description of the design. This is comprised of conceptual design layout and assembly requirements. [STEP-203]

F.1.11 **change request*** – request to change a previously established requirement or set of requirements, due to a non-conformance or a change in insights, technical or financial constraints, users’ desires.

F.1.12 **Define and Formulate Product Design** – the process of making a complete product design definition. This process includes preliminary and detailed design. It is supported by functional and performance assessments and analyses, design simulations, alternate design evaluations, trade-offs between alternate design solutions, breadboard, development and qualification testing.

F.1.13 **Deploy and Operate Product*** – the process of putting into work and thereafter operating a delivered product. This will mostly be done by another party than the product developers. In this AP the relevance of this activity is that it may produce operation results which one wants to compare to analysis predictions for design evaluation purposes.

F.1.14 **design ideas*** – the ideas for development approaches and design solutions in order to fulfill the form, fit, function and performance requirements for a suggested product. Partly from [STEP-203].

F.1.15 **design feedback** – for a particular product development.

F.1.16 **design knowledge*** – existing engineering knowledge derived from education, training and experience. This comprises formal or informal, structured, explicit or implicit knowledge of previous design, analysis and verification approaches, of standards, procedures, parts and materials, of methods and tools, in other words all knowledge which is relevant for a particular product development.

F.1.17 **Develop Product** – the process of developing, i.e. designing, analysing and testing a product.

F.1.18 **Develop, Produce, Operate Space Product** - top level activity for this AAM, comprising the complete life cycle of a space product from conception to operation.

F.1.19 **Do Thermal Network Analysis** – the process of performing thermal network analysis in which the heatbalances for all nodes in a thermal lumped parameter model or all elements in a thermal finite element model are solved for steady state or transient analysis cases. The main results from this activity are temperatures for all nodes or elements at requested analysis run times.

F.1.20 **Do Thermal Radiative Analysis** – the process of performing thermal radiative analysis in which radiative exchange factors and environmental fluxes are computed for a given thermal radiative model and a given mission.

F.1.21 **Establish Product Requirements*** – the process of establishing the requirements for a particular product. This is an iterative process in which the product design, functional and performance requirements are initially derived from the user requirements, then refined stepwise to gradually increasing detail, updated for non-conformances or change requests.

F.1.22 **Manage Project*** – the process of managing the project, which includes directing the project team, creating and maintaining schedules and budget plans, allocating human and other resources, handling of financial and contractual matters, keeping the liaison with external parties (customer, sub-contractors, etc.).

F.1.23 **material databases** – databases with material descriptions, physical properties, method and range of application, qualification status, etc.

F.1.24 **mission definition** – definition of the aspects of the mission to be fullfilled by a space product that are relevant to its development. The mission aspects include definitions of: orbit trajectories, attitudes, natural thermal environment, operational modes.

F.1.25 **non-conformance** – the discrepancy between a product requirement and a predicted or achieved performance.

F.1.26 **operating product*** – accepted and delivered product in operational state.

F.1.27 **operation constraints*** – constraints and/or requirements originating from the product development stage to be fulfilled in the operational situation, e.g. attitude constraints for manoeuvres in order to avoid extreme thermal loads on a spacecraft.

F.1.28 **operation results** – the (measured) results obtained from operating a product in a particular operational situation.

F.1.29 **parts databases** – databases with descriptions, physical properties, method and range of application, qualification status, etc. of standard parts and components.

F.1.30 **preconditions*** – conditions and information pertaining the a certain product development that exist prior to the start of the development, e.g. market research information.

F.1.31 **Produce Product*** – the process of actually producing (i.e. manufacturing, assembling, integrating and acceptance testing) one or more units of the product according to the design description and specifications established in the Develop Product activity.

F.1.32 **product description** – all data needed to describe a product, e.g. geometric information of the shapes, the product assembly tree, tolerances, materials to be used, processes to be applied, surface treatments.

F.1.33 **production equipment**– tools and machines needed for the production of the product of interest, e.g. drills, milling machines, lathes, autoclaves, thermal blanket cutting, sewing and fitting tools, product dedicated tools, etc.

F.1.34 **production facilities** – facilities for the production of the product of interest, e.g. assembly lines, clean rooms.

F.1.35 **production results** – the (measured) results obtained during the Produce Product activity, e.g. the measurements performed during acceptance testing.

F.1.36 **project directives*** – the directives resulting from the Manage Project activity. The directives are a translation of the schedule, budget, staffing and other resource allocation constraints with the objective of effectuating the best achievable control of the Develop Product and Produce Product activities.

F.1.37 **raw material*** – a crude or partially processed material used as a feedstock for a processing operation [STEP-214].

F.1.38 **requirements*** – verifiable criteria that specify the functionality and/or characteristics and/or performance of (1) the product to be developed, (2) the methods to be used, (3) the tools to be used.

F.1.39 **research papers*** – publications of research (e.g. conference papers or magazine articles) describing the approaches taken, the developed methodologies and/or the obtained results during a product development or operation activity.

F.1.40 **resource budgets*** – technical budgets for e.g. mass, power consumption or allowable (thermal) distortion which must be complied with or which should be taken as targets during Define and Formulate Product Design activities. The resource budgets may be broken down per engineering discipline.

F.1.41 **schedule constraints*** – constraints imposing the schedule for the product development.

F.1.42 **standard part*** – a partially processed material used as a feedstock for a processing operation [STEP-214]. Mostly a component-like item available for selection from some parts catalogue.

F.1.43 **Synthesise and Refine Configuration*** – the process of ensuring the form, fit, function and performance from one section of the design into the overall product. This is a systems engineering activity which defines the product structure, allocates functions to different sections of the design, sets resource budgets and in general coordinates and integrates the designs from the participating engineering disciplines.

F.1.44 **test case*** – definition of a case to be tested, comprising description of the required test set-up and/or test facility, definition of the test script, as well as definition and assignment of all relevant test parameters.

F.1.45 **test model*** – a representation of (part of) the product under development – mostly in hardware – for a particular test.

F.1.46 **test facilities**– facilities where development, qualification and/or acceptance test runs can be performed, e.g. cooled vacuum test chambers with solar simulator.

F.1.47 **test results** – the results obtained from a test run with a particular test model for a particular test case.

F.1.48 **tooling requirements*** – requirements for tools needed in a particular product's production process.

F.1.49 **user requirements*** – product requirements specified by the customer and/or targeted user(s).

F.2 Application Activity Model Diagrams

Below the AAM diagrams are presented in IDEF0 format. For a large part they take into account the general engineering process for the development of space products.

In order to help understand the diagrams, below a concise illustration of the “activity”, the main element of the IDEF0 formalism, is presented.

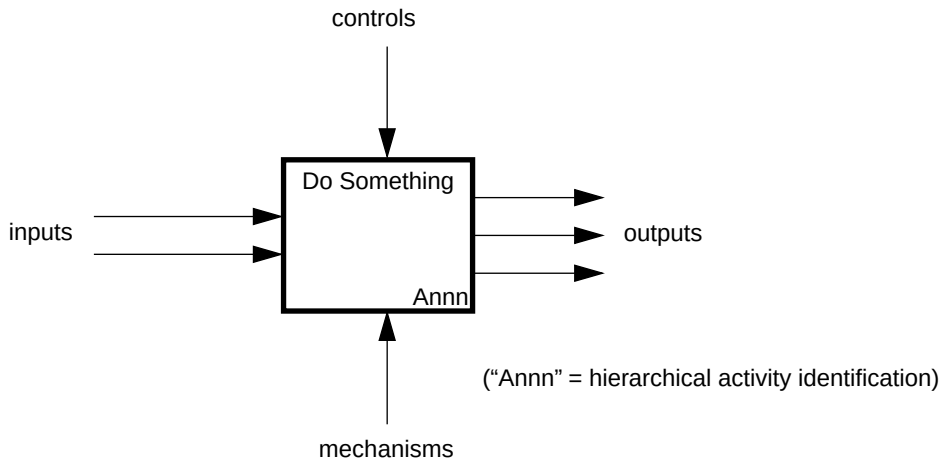


Figure F–1 - The graphical representation of a IDEF0 activity

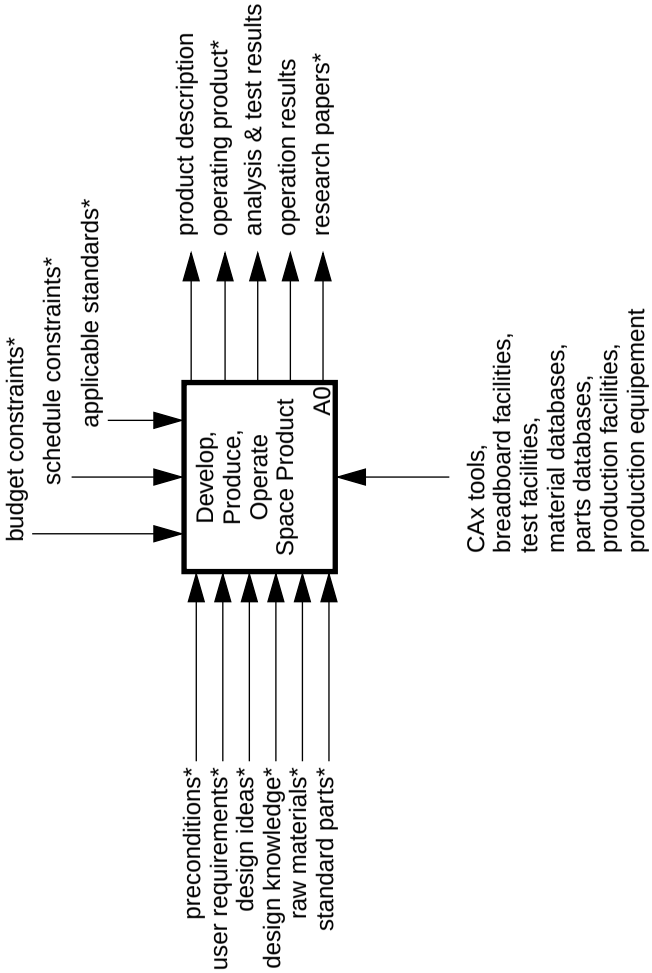


Figure F–2 - Top level process for development, production and operation of a space product

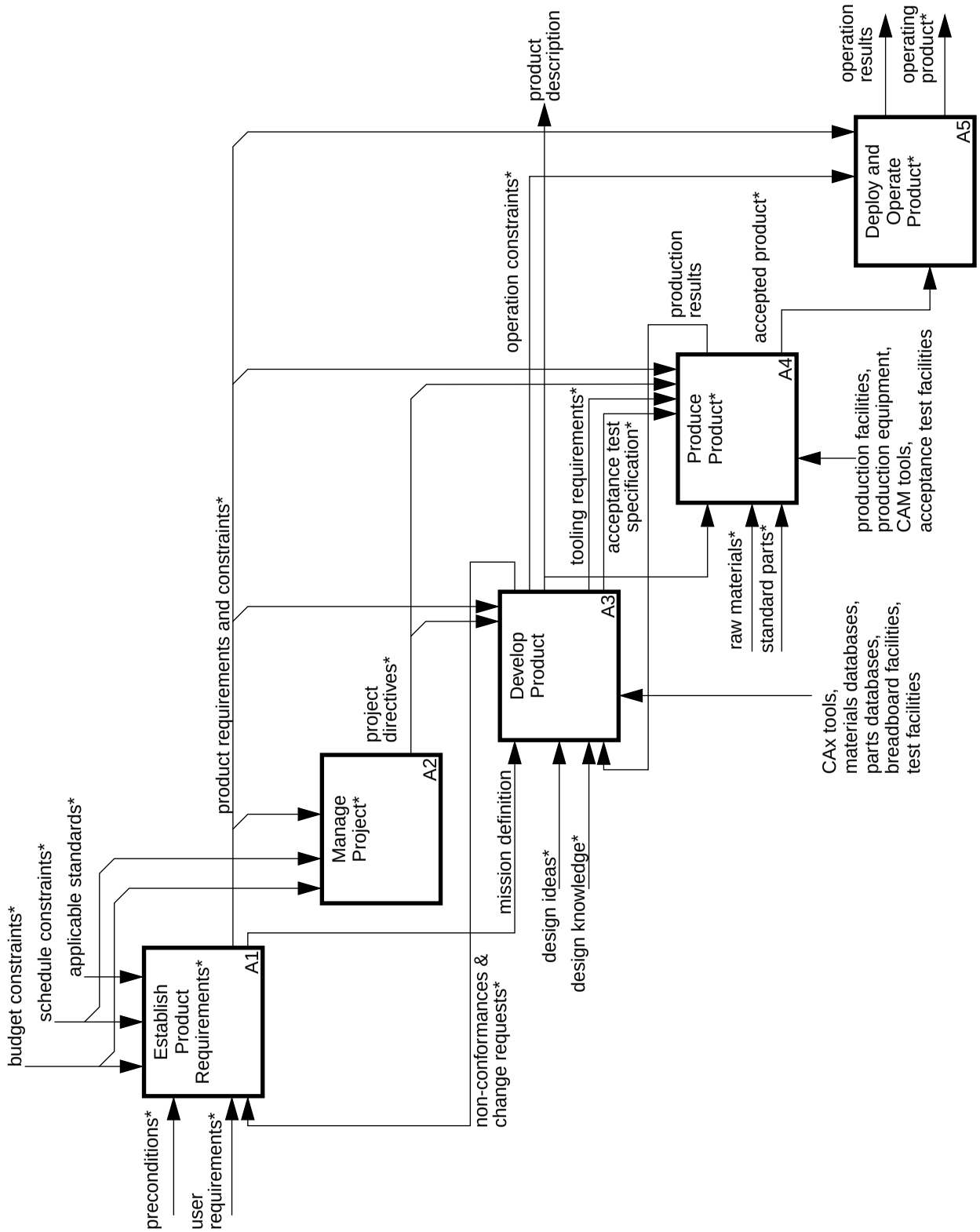


Figure F–3 - Breakdown of process A0: Develop, Produce and Operate Space Product

NOTE: STEP-TAS will consider mainly A3: Develop Product

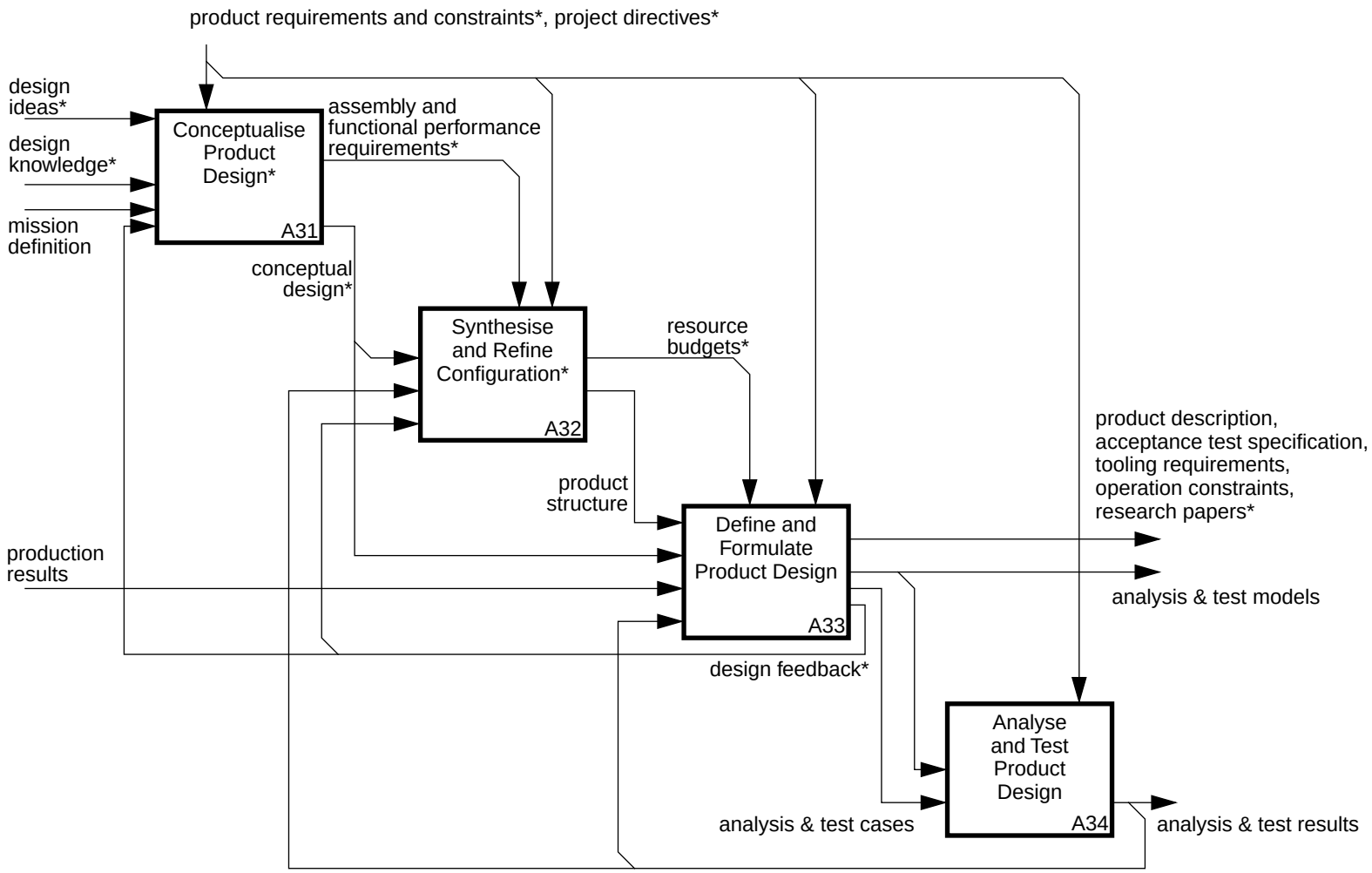


Figure F–4 - Breakdown of process A3: Develop Product

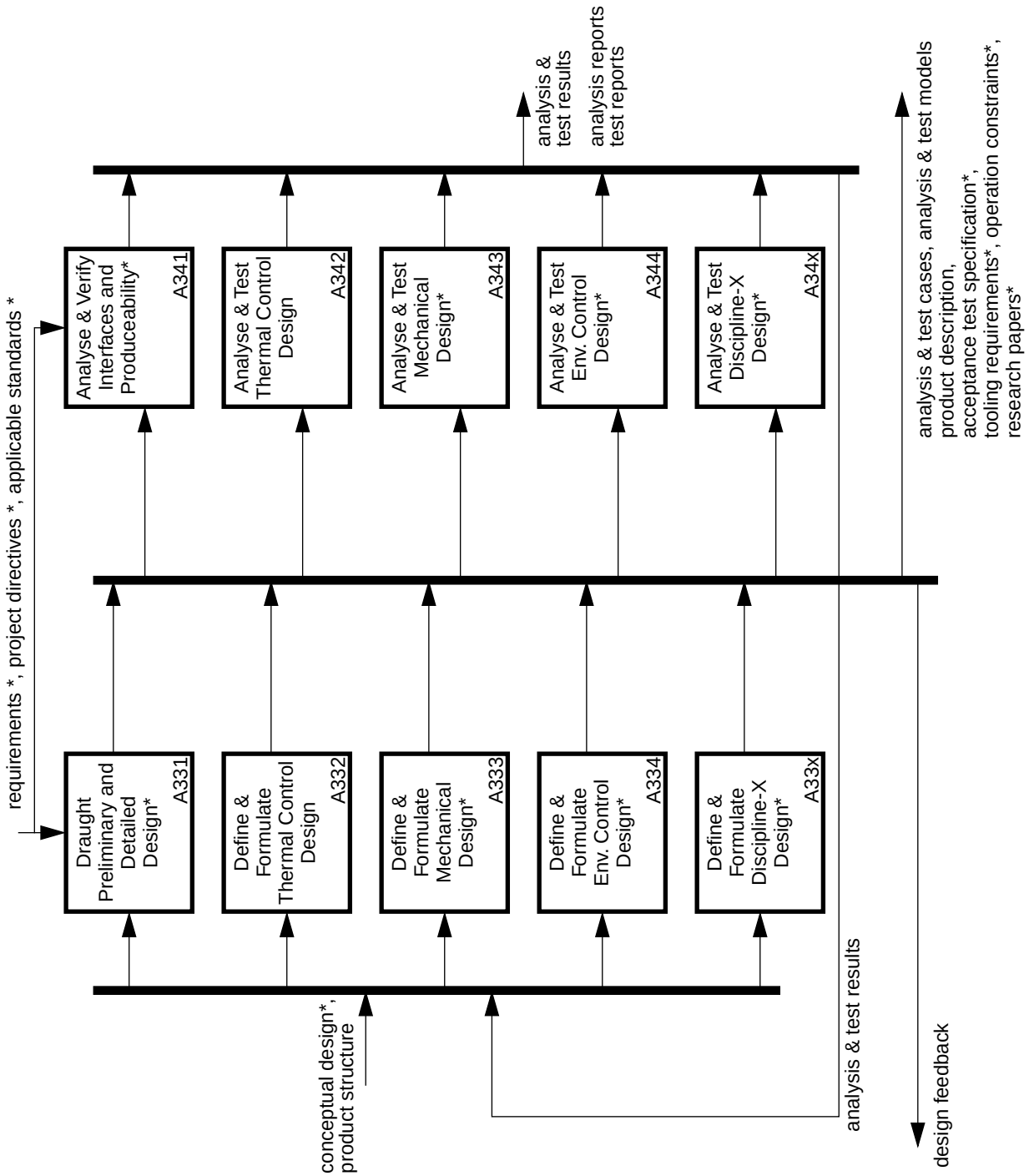


Figure F–5 - Breakdown into disciplines of processes A33 "Define and Formulate Design" and A34 "Analyse and Test Design" in a *concurrent engineering* perspective

NOTE 1: Mission definition and analysis are assumed to be included in activities A1, A31 and A32.

NOTE 2: Strictly this diagram is not correct IDEF0. Here an attempt is made to show the highly concurrent engineering processes that operate on shared sets of data, which are depicted as "data buses".

NOTE 3: "Discipline-X" is used to indicate any other discipline that might be involved in the development of a space product, e.g. Electrical Engineering, Attitude Control, Propulsion, Optical Engineering, Telecommunication, Software Engineering, Ergonomics, FMECA, FDIR, RAMS, and so forth.

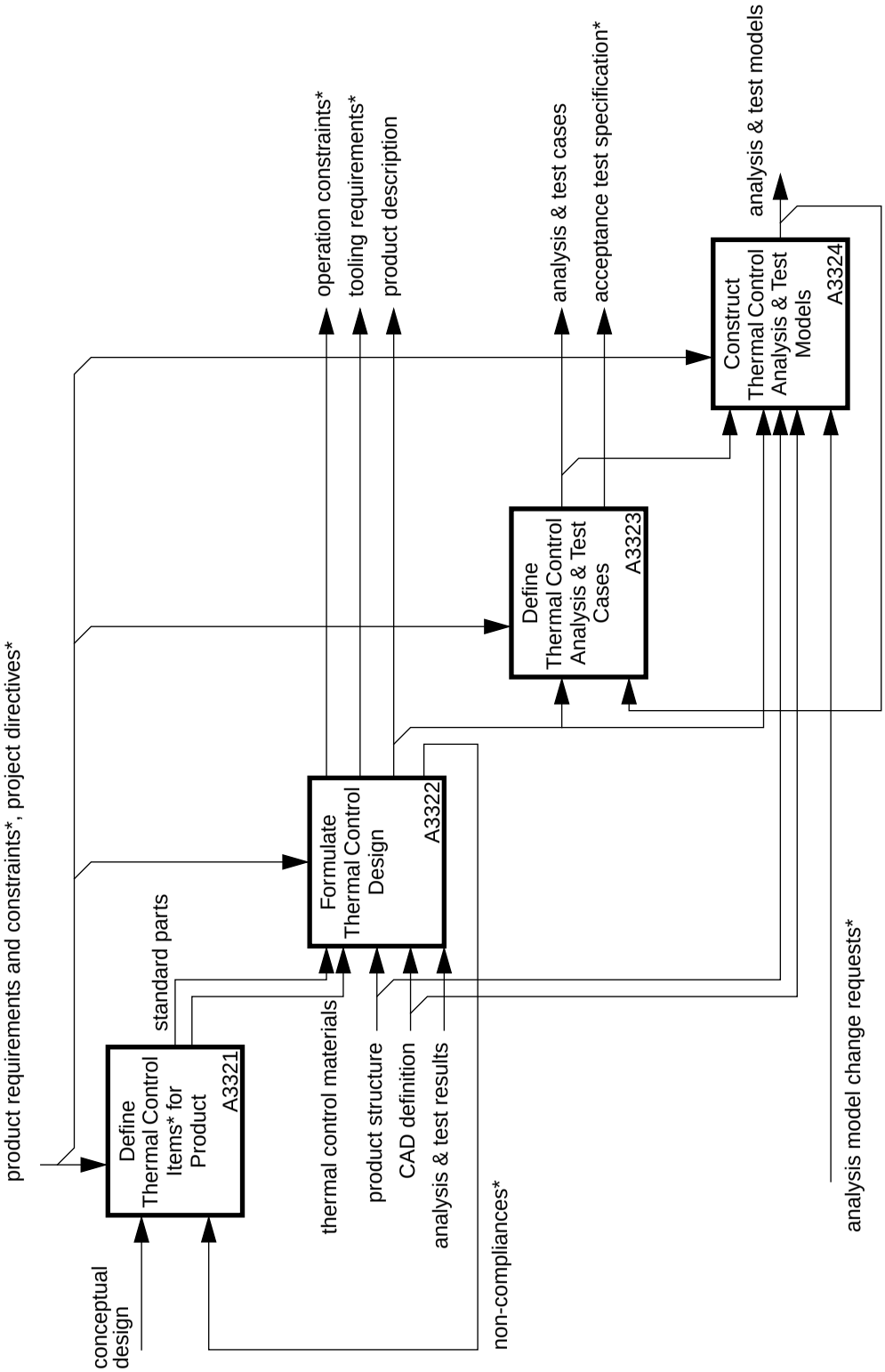


Figure F–6 - Breakdown of process A332: Define and Formulate Thermal Control Design

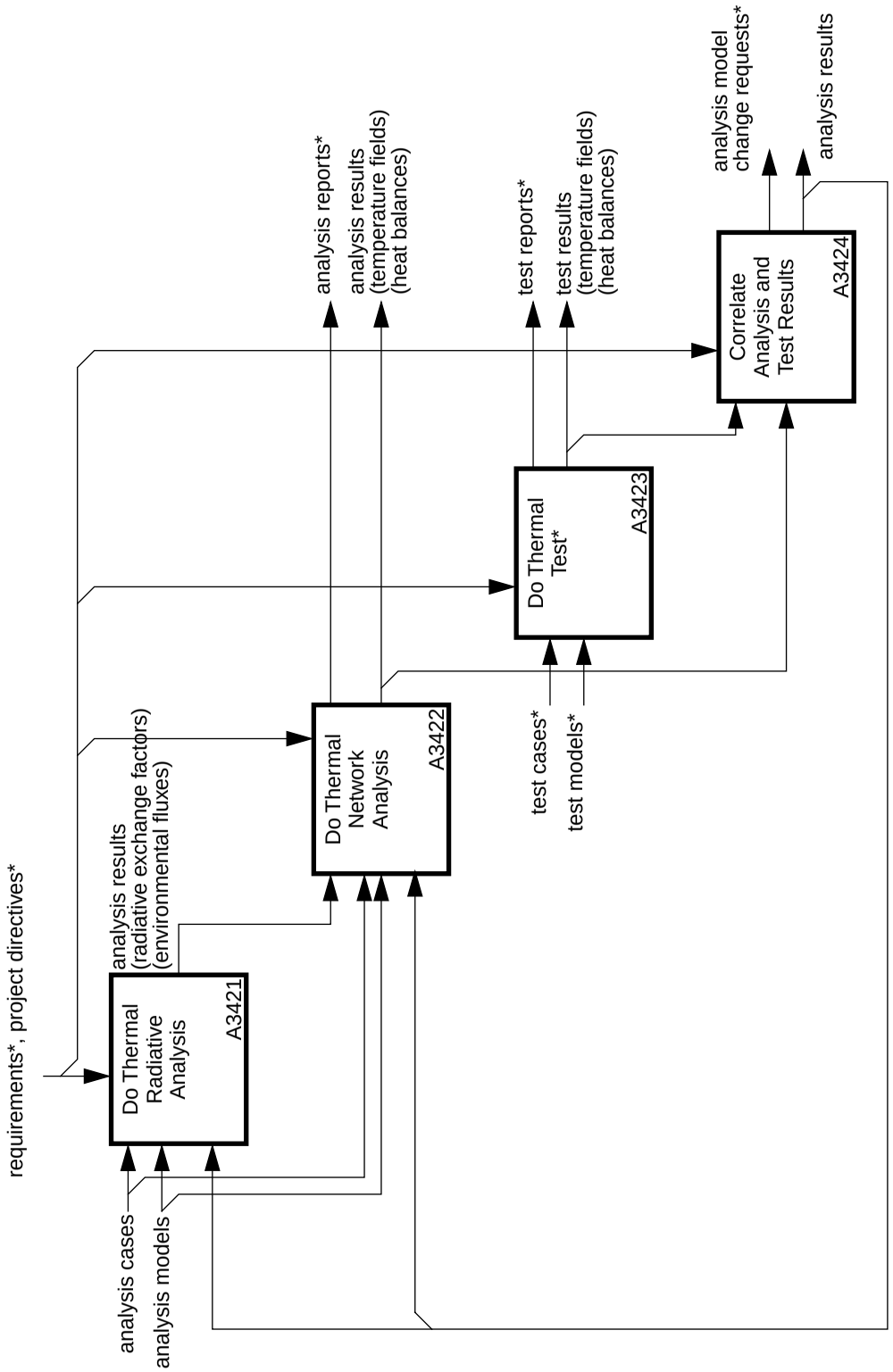


Figure F–7 - Breakdown of process A342: Analyse and Test Thermal Control Design

Annex G - Application Reference Model (Informative)

G.1 EXPRESS Definition of ARM Application Objects

```
SCHEMA tas_arm_schema;

-- Checked with ST-EXPRESS by STEPTools Inc.
-- and IS versions of EXPRESS schemas for ISO 10303 parts 41, 42, 43

(*
-------------------------------------------------------------------------------
Revision record (release, date, author, description)
-------------------------------------------------------------------------------
2.4.0, 28-Mar-96, HPdK
Baseline release with STEP-TAS AP r2.4

2.4.1, 05-May-97, HPdK
1) Resolved circular reference on ato_case.abscissa_class is resolved by including
   nrf_arm_schema r1.4.1.
2) Added geometric_submodel_usage as a possibe type in boolean_first_operand_type.

-------------------------------------------------------------------------------
*)

-- Resource ISO 10303-42:1994
REFERENCE FROM geometry_schema (
  cartesian_point,
  direction );

(* Here follows the included nrf_arm_schema r1.4.1 as a resource *)

USE FROM support_resource_schema (
  identifier,
  label,
  text );                                       -- defined in ISO 10303 part 41

USE FROM date_time_schema;
                                                -- defined in ISO 10303 part 41

USE FROM measure_schema;
                                                -- defined in ISO 10303 part 41

USE FROM person_organization_schema;
                                                -- defined in ISO 10303 part 41

USE FROM document_schema;
                                                -- defined in ISO 10303 part 41

USE FROM security_classification_schema (
  security_classification_level );              -- defined in ISO 10303 part 41

TYPE abscissa_sequencing_type = ENUMERATION OF (
  strictly_decreasing,
  monotonic_decreasing,
  monotonic_increasing,
  strictly_increasing);
END_TYPE;

(* address used from ISO-10303-part-41 person_organization_schema
 * ENTITY address;
 *   internal_location       : OPTIONAL label;
 *   street_number           : OPTIONAL label;
 *   street                  : OPTIONAL label;
 *   postal_box              : OPTIONAL label;
 *   town                    : OPTIONAL label;
```

```
 *   region                  : OPTIONAL label;
 *   postal_code             : OPTIONAL label;
 *   country                 : OPTIONAL label;
 *   facsimile_number        : OPTIONAL label;
 *   telephone_number        : OPTIONAL label;
 *   electronic_mail_address : OPTIONAL label;
 *   telex_number            : OPTIONAL label;
 * WHERE
 *   WR1: EXISTS(internal_location)       OR
 *        EXISTS(street_number)           OR
 *        EXISTS(street)                  OR
 *        EXISTS(postal_box)              OR
 *        EXISTS(town)                    OR
 *        EXISTS(region)                  OR
 *        EXISTS(postal_code)             OR
 *        EXISTS(country)                 OR
 *        EXISTS(facsimile_number)        OR
 *        EXISTS(telephone_number)        OR
 *        EXISTS(electronic_mail_address) OR
 *        EXISTS(telex_number);
 * END_ENTITY;
 *)

ENTITY approval;
  -- is a simplification of approval in ISO-10303-41 approval_schema
  level                  : label;
  date_time              : date_and_time;
  by_person_organization : person_and_organization;
END_ENTITY;

ENTITY ato_aspect;
  name                   : label;
  for_property_class     : property_class_usage;
  lifetime               : lifetime_type;
  component_sequence     : model_component_sequence;
  scans                  : LIST [1:?] OF scan;
  security_class         : OPTIONAL security_classification_level;
END_ENTITY;

ENTITY ato_campaign;
  -- "ato" stands for "analysis, test or operation"
  of_project                      : organizational_project;
  name                            : label;
  description                     : text;
  creation_date_and_time          : date_and_time ;
  last_modification_date_and_time : OPTIONAL date_and_time ;
  contact_person_organization     : person_and_organization;
  approvals                       : SET [0:?] OF approval;
  security_class                  : OPTIONAL security_classification_level;
  arm_schema_name                 : STRING;
  arm_protocol_year               : INTEGER;
UNIQUE
  ur1 : of_project, name;
END_ENTITY;

ENTITY ato_case;
  -- "ato" stands for "analysis, test or operation"
  of_campaign         : ato_campaign;
  id                  : identifier;
  name                : label;
  description         : text;
  root_model          : network_model;
  abscissa_class      : abscissa_property_class_usage;
  abscissa_sequencing : OPTIONAL abscissa_sequencing_type;
  predefined_aspects  : SET [0:?] OF ato_aspect;
  security_class      : OPTIONAL security_classification_level;
INVERSE
  phases              : SET [0:?] OF ato_phase FOR of_case;
```

```
  unit_assignments    : SET [1:?] OF unit_assignment FOR of_case;
UNIQUE
  ur1 : of_campaign, id;
END_ENTITY;

ENTITY ato_phase;
  -- "ato" stands for "analysis, test or operation"
  of_case            : ato_case;
  id                 : identifier;
  name               : label;
  description        : text;
  predefined_aspects : SET [0:?] OF ato_aspect;
  security_class     : OPTIONAL security_classification_level;
UNIQUE
  ur1 : of_case, id;
END_ENTITY;

(* calendar_date used from ISO-10303-41 date_time_schema
 * ENTITY calendar_date
 *  SUBTYPE OF (date);
 *   day_component   : day_in_month_number;
 *   month_component : month_in_year_number;
 * WHERE
 *   WR1: valid_calendar_date (SELF);
 * END_ENTITY;
 *)

ENTITY cyclic_tabular_function
  SUBTYPE OF ( tabular_function );
  period : scalar_value;
WHERE
  wr1 : SIZEOF( SELF\parameterized_function.parameter_properties ) = 1;
END_ENTITY;

(* date used from ISO-10303-41 date_time_schema
 * ENTITY date
 *   SUPERTYPE OF (ONEOF (calendar_date,
 *                        ordinal_date,
 *                        week_of_year_and_day_date));
 *   year_component : year_number;
 * END_ENTITY;
 *)

(* date_and_time used from ISO-10303-41 date_time_schema
 * ENTITY date_and_time;
 *  date_component : date;
 *  time_component : local_time;
 * END_ENTITY;
 *)

(* document used from ISO-10303-41 document_schema
 * ENTITY document;
 *   id          : identifier;
 *   name        : label;
 *   description : text;
 *   kind        : document_type;
 * UNIQUE
 *   UR1: id;
 * END_ENTITY;
 *)

(* document_type used from ISO-10303-41 document_schema
 * ENTITY document_type;
 *   name : label; -- Modified! Attribute was called product_data_type in ISO-
10303-41
 * END_ENTITY;
 *)
```

```
(* document_usage_constraint type used from ISO-10303-41 document_schema
 * ENTITY document_usage_constraint;
 *   source                : document;
 *   subject_element       : label;
 *   subject_element_value : text;
 * END_ENTITY;
 *)

TYPE lifetime_type = ENUMERATION OF (
  invariant,
  sample,
  interval ) ;
END_TYPE;

ENTITY list_of_descriptive_values;
  applicable_for : SET [1:?] OF property_class_usage;
  name           : label;
  entries        : LIST [1:?] OF STRING;
END_ENTITY;

ENTITY list_of_functions;
  applicable_for : SET [1:?] OF property_class_usage;
  name           : label;
  entries        : LIST [1:?] OF parameterized_function;
END_ENTITY;

(* local_time used from ISO-10303-part-41 date_time_schema
 * ENTITY local_time;
 *   hour_component   : hour_in_day;
 *   minute_component : OPTIONAL minute_in_hour;
 *   second_component : OPTIONAL second_in_minute;
 *   zone             : coordinated_universal_time_offset;
 * WHERE
 *   WR1: valid_time (SELF);
 * END_ENTITY;
 *)

ENTITY model_component_sequence;
  id         : identifier;
  name       : label;
  components : LIST [1:?] OF UNIQUE network_model_component;
DERIVE
  root_model : network_model := find_root_model( components );
END_ENTITY;

FUNCTION find_root_model( mcl : LIST OF network_model_component ) : network_model;
-- to be implemented
  RETURN([]);
END_FUNCTION;

ENTITY model_represents_product_relationship;
  model_constituent   : network_model_or_constituent;
  represented_product : product_definition;
END_ENTITY;

ENTITY network_model;
  id             : identifier;
  version_id     : identifier;
  name           : label;
  description    : text;
  security_class : OPTIONAL security_classification_level;
  constituents   : SET [1:?] OF network_model_constituent;
UNIQUE
  ur1: id, version_id;
END_ENTITY;

TYPE network_model_component = SELECT (
  network_model,
```

```
  network_node,
  network_node_relationship,
  submodel_usage,
  network_node_usage,
  network_node_relationship_usage );
END_TYPE;

TYPE network_model_constituent = SELECT (
  network_node,
  network_node_relationship,
  submodel_usage );
END_TYPE;

TYPE network_model_or_constituent = SELECT (
  network_model,
  network_model_constituent );
END_TYPE;

ENTITY network_node;
  id             : identifier;
  name           : OPTIONAL label;
  class          : LIST [1:?] OF network_node_class;
  security_class : OPTIONAL security_classification_level;
END_ENTITY;

ENTITY network_node_class;
  name  : label;
UNIQUE
  ur1: name;
END_ENTITY;

TYPE network_node_select = SELECT (
  network_node,
  network_node_usage );
END_TYPE;

ENTITY network_node_relationship;
  id             : identifier;
  name           : OPTIONAL label;
  class          : LIST [1:?] OF network_node_relationship_class;
  nodes          : LIST [2:?] OF network_node_select;
  security_class : OPTIONAL security_classification_level;
END_ENTITY;

ENTITY network_node_relationship_class;
  name  : label;
UNIQUE
  ur1: name;
END_ENTITY;

ENTITY network_node_relationship_usage;
  used_submodel : submodel_usage;
  relationship  : network_node_relationship;
END_ENTITY;

ENTITY network_node_usage;
  used_submodel : submodel_usage;
  node          : network_node;
END_ENTITY;

(* organization used from ISO-10303-41 person_organization_schema
 * ENTITY organization;
 *   id          : OPTIONAL identifier;
 *   name        : label;
 *   description : text;
 * END_ENTITY;
 *)
```

```
(* organizational_address used from ISO-10303-41 person_organization_schema
 * ENTITY organizational_address
 *   SUBTYPE OF (address);
 *   organizations : SET [1:?] OF organization;
 *   description   : text;
 * END_ENTITY;
 *)

(* organizational_project used from ISO-10303-41 person_organization_schema
 * ENTITY organizational_project;
 *   name                      : label;
 *   description               : text;
 *   responsible_organizations : SET[1:?] OF organization;
 * END_ENTITY;
 *)

ENTITY parameterized_function
  ABSTRACT SUPERTYPE OF( ONEOF(
    parameterized_function_with_formula,
    polynomial_function,
    tabular_function ) );
  name                 : label;
  description          : text;
  parameter_properties : LIST [0:?] OF property_class_scalar_quantitative;
  result_property      : property_class_scalar_quantitative;
  source               : OPTIONAL document_usage_constraint;
  definition_timestamp : OPTIONAL date_and_time;
END_ENTITY;

ENTITY parameterized_function_with_formula
  SUBTYPE OF( parameterized_function );
  formula                  : text;
  language                 : OPTIONAL label;
  creator_tool_or_facility : OPTIONAL label;
END_ENTITY;

(* person used from ISO-10303-41 person_organization_schema
 * ENTITY person;
 *   id            : identifier;
 *   last_name     : OPTIONAL label;
 *   first_name    : OPTIONAL label;
 *   middle_names  : OPTIONAL LIST [1:?] OF label;
 *   prefix_titles : OPTIONAL LIST [1:?] OF label;
 *   suffix_titles : OPTIONAL LIST [1:?] OF label;
 * UNIQUE
 *   UR1: id;
 * WHERE
 *   WR1: EXISTS(last_name) OR EXISTS(first_name);
 * END_ENTITY;
 *)

(* person_and_organization used from ISO-10303-41 person_organization_schema
 * ENTITY person_and_organization;
 *   the_person       : person;
 *   the_organization : organization;
 * END_ENTITY;
 *)

(* personal_address used from ISO-10303-41 person_organization_schema
 * ENTITY personal_address
 *   SUBTYPE OF (address);
 *   people      : SET [1:?] OF person;
 *   description : text;
 * END_ENTITY;
 *)

ENTITY polynomial_function
  SUBTYPE OF( parameterized_function );
```

```
  degree                : INTEGER;
  coefficients          : LIST [1:?] OF REAL;
  lower_domain_bounds   : ARRAY [1:3] OF OPTIONAL REAL;
  upper_domain_bounds   : ARRAY [1:3] OF OPTIONAL REAL;
WHERE
  wr1: (degree >= 1) AND (degree <= 10);
END_ENTITY;

ENTITY product;
  id                  : identifier;
  name                : label;
  description         : text;
  frame_of_reference : SET [1:?] OF product_context;
END_ENTITY;

ENTITY product_context;
  name            : label;
  discipline_type : label;
END_ENTITY;

ENTITY product_definition;
  id                  : identifier;
  description         : text;
  frame_of_reference : product_definition_context;
UNIQUE
  ur1: id, frame_of_reference;
END_ENTITY;

ENTITY product_definition_context;
  name              : label;
  life_cycle_stage : label;
END_ENTITY;

ENTITY product_next_assembly_usage_relationship;
  id          : identifier;
  assembly    : product_definition;
  constituent : product_definition;
UNIQUE
  ur1: id, assembly;
END_ENTITY;

ENTITY product_version;
  of_product  : product;
  id          : identifier;
  description : text;
  definitions : SET [1:?] OF product_definition;
UNIQUE
  ur1: id, of_product;
END_ENTITY;

TYPE property_class = SELECT (
  property_class_scalar,
  property_class_tensor );
END_TYPE;

ENTITY property_class_scalar
  ABSTRACT SUPERTYPE OF ( ONEOF (
    property_class_scalar_descriptive,
    property_class_scalar_functional,
    property_class_scalar_quantitative ) );
  name           : property_name;
  symmetry       : OPTIONAL property_symmetry;
END_ENTITY;

ENTITY property_class_scalar_descriptive
  SUBTYPE OF ( property_class_scalar );
DERIVE
  value_container : SET [1:?] OF list_of_descriptive_values
```

```
    := find_list_of_descriptive_values( SELF );
END_ENTITY;

FUNCTION find_list_of_descriptive_values(
  pcsd : property_class_scalar_descriptive ) : SET OF list_of_descriptive_values;
-- to be implemented
  RETURN([]);
END_FUNCTION;

ENTITY property_class_scalar_functional
  SUBTYPE OF ( property_class_scalar );
DERIVE
  function_container : list_of_functions
    := find_list_of_functions( SELF );
END_ENTITY;

FUNCTION find_list_of_functions(
  pcsf : property_class_scalar_functional ) : SET OF list_of_functions;
-- to be implemented
  RETURN([]);
END_FUNCTION;

ENTITY property_class_scalar_quantitative
  SUBTYPE OF ( property_class_scalar );
END_ENTITY;

ENTITY property_class_tensor;
  name           : property_name;
  tensor_order   : INTEGER;
  dimensionality : INTEGER;
INVERSE
  members        : SET [1:?] OF scalar_in_tensor FOR tensor;
WHERE
  wr1: (   (tensor_order = 1) AND (dimensionality >= 2)      ) OR
       ( ( (tensor_order = 2) AND (dimensionality =  2) ) OR
         ( (tensor_order = 2) AND (dimensionality =  3) )    ) OR
       ( ( (tensor_order = 4) AND (dimensionality =  2) ) OR
         ( (tensor_order = 4) AND (dimensionality =  3) )    ) ;
END_ENTITY;

ENTITY property_class_usage;
  for_case : ato_case;
  property : property_class;
  meaning  : LIST [1:?] OF property_role;
END_ENTITY;

(* created per release 1.4.1 *)
ENTITY abscissa_property_class_usage
  SUBTYPE OF (property_class_usage);
INVERSE
  for_case : ato_case FOR abscissa_class;
END_ENTITY;

ENTITY property_name;
  name : label;
UNIQUE
  ur1 : name;
END_ENTITY;

ENTITY property_role;
  name : label;
UNIQUE
  ur1 : name;
END_ENTITY;

TYPE property_symmetry = ENUMERATION OF (
  symmetrical,
  antisymmetrical );
```

```
END_TYPE;

ENTITY run;
  of_phase                 : ato_phase;
  id                       : identifier;
  name                     : label;
  description              : text;
  timestamp                : date_and_time;
  creator_tool_or_facility : OPTIONAL label;
  aspects                  : SET [1:?] OF ato_aspect;
UNIQUE
  ur1: of_phase, id;
END_ENTITY;

ENTITY run_sequence;
  of_case     : ato_case;
  id          : identifier;
  name        : label;
  description : text;
  runs        : LIST [1:?] OF run;
UNIQUE
  ur1 : of_case, id;
END_ENTITY;

ENTITY scalar_in_tensor;
  tensor   : property_class_tensor;
  scalar   : property_class_scalar;
  position : INTEGER;
  roles    : LIST [1:?] OF property_role;
END_ENTITY;

TYPE scalar_value = NUMBER ;
END_TYPE;

ENTITY scan
  ABSTRACT SUPERTYPE OF( ONEOF(
    scan_of_sampled_values,
    scan_of_derived_values ) );
  abscissa_value   : OPTIONAL scalar_value;
  mask             : OPTIONAL scan_mask;
  number_of_values : INTEGER;
  values           : ARRAY [1:number_of_values] OF OPTIONAL scalar_value;
INVERSE
  of_aspect        : ato_aspect FOR scans;
END_ENTITY;

ENTITY scan_derivation_procedure;
  name : label;
UNIQUE
  ur1 : name;
END_ENTITY;

ENTITY scan_mask;
  defined_ranges : LIST [1:?] OF ARRAY [1:2] OF INTEGER;
END_ENTITY;

ENTITY scan_of_derived_values
  SUBTYPE OF ( scan );
  derivation_procedure : scan_derivation_procedure;
  abscissa_start       : OPTIONAL scalar_value;
  abscissa_end         : OPTIONAL scalar_value;
END_ENTITY;

ENTITY scan_of_sampled_values
  SUBTYPE OF ( scan );
WHERE
  wr1: EXISTS( SELF\scan.abscissa_value );
END_ENTITY;
```

```
(* security_classification_level used from
 * ISO 10303-21 security_classification_schema
 *
 * ENTITY security_classification_level;
 *   name : label;
 * END_ENTITY;
 *)

(*
 * restrict_security_classification_level copied from ISO-10303-203
 *)
RULE restrict_security_classification_level FOR (security_classification_level);
WHERE
  WR1: SIZEOF (QUERY (scl <* security_classification_level |
       NOT (scl.name IN ['unclassified', 'classified', 'proprietary',
       'confidential', 'secret', 'top_secret']))) = 0;
END_RULE; -- restrict_security_classification_level

ENTITY submodel_usage;
  id          : identifier;
  name        : label;
  submodel    : network_model;
  source_node : network_node;
  target_node : network_node;
END_ENTITY;

ENTITY tabular_function
  SUBTYPE OF ( parameterized_function );
  parameter_values     : LIST [1:?] OF scalar_value;
  function_values      : LIST [1:?] OF scalar_value;
  interpolation_type   : tabular_interpolation_type;
  interpolation_degree : OPTIONAL INTEGER;
END_ENTITY;

TYPE tabular_interpolation_type = ENUMERATION OF (
  linear_logarithmic,
  polynomial );
END_TYPE;

(* unit used from ISO-10303-41 measure_schema
 * Here some additional ENTITY definitions are given to provide
 * insight on how units are defined in ISO-10303-41.
 *
 * TYPE unit = SELECT
 *   (named_unit,
 *    derived_unit);
 * END_TYPE;
 *
 * ENTITY named_unit
 *   SUPERTYPE OF (ONEOF (si_unit, conversion_based_unit, context_dependent_unit)
 *                 ANDOR
 *                 ONEOF (length_unit,
 *                        mass_unit,
 *                        time_unit,
 *                        electric_current_unit,
 *                        thermodynamic_temperature_unit,
 *                        amount_of_substance_unit,
 *                        luminous_intensity_unit,
 *                        plane_angle_unit,
 *                        solid_angle_unit,
 *                        area_unit,
 *                        volume_unit,
 *                        ratio_unit ));
 *   dimensions : dimensional_exponents;
 * END_ENTITY;
 *
 * ENTITY derived_unit;
```

```
 *   elements : SET [1:?] OF derived_unit_element;
 * WHERE
 *   WR1 : ( SIZEOF ( elements ) > 1 ) OR
 *           (( SIZEOF ( elements ) = 1 ) AND ( elements[1].exponent <> 1.0 ));
 * END_ENTITY;
 *
 * ENTITY derived_unit_element;
 *   unit     : named_unit;
 *   exponent : REAL;
 * END_ENTITY;
 *)

ENTITY unit_assignment;
  of_case            : ato_case;
  for_property_class : property_class_scalar;
  assigned_unit      : unit;
UNIQUE
  ur1: of_case, for_property_class;
END_ENTITY;

(* Here ends the included nrf_arm_schema r1.4.1 *)

(* start of specific TAS part of schema *)

ENTITY space_mission
  SUBTYPE OF ( ato_campaign );
END_ENTITY;

ENTITY space_mission_case
  SUBTYPE OF ( ato_case );
  coordinate_system : space_coordinate_system;
END_ENTITY;

ENTITY space_mission_phase
  SUBTYPE OF ( ato_phase );
  start_event   : space_mission_event;
  end_event     : space_mission_event;
  space_aspects : SET [1:?] OF space_mission_aspects;
END_ENTITY;

ENTITY space_mission_aspect
  ABSTRACT SUPERTYPE OF ( ONEOF (
    orbit_arc,
    main_body_placement,
    kinematic_articulation,
    material_property_environment,
    thermal_environment ) );
  of_case     : space_mission_case;
  id          : identifier;
  name        : label;
  description : text;
UNIQUE
  ur1: of_case, id;
END_ENTITY;

ENTITY space_misson_event
  SUPERTYPE OF ( ONEOF (
    space_mission_event_with_epoch,
    space_mission_event_with_running_time,
    space_mission_event_with_true_anomaly ) );
  of_case     : space_mission_case;
  id          : identifier;
  name        : label;
  description : text;
UNIQUE
  ur1: of_case, id;
END_ENTITY;
```

```
ENTITY space_mission_event_with_epoch
  SUBTYPE OF ( space_misson_event );
  epoch : date_and_time;
END_ENTITY;

ENTITY space_mission_event_with_running_time
  SUBTYPE OF ( space_misson_event );
  running_time : time_measure;
END_ENTITY;

ENTITY space_mission_event_with_true_anomaly
  SUBTYPE OF ( space_misson_event );
  true_anomaly : plane_angle_measure;
END_ENTITY;

TYPE space_coordinate_system_type = ENUMERATION OF (
  star_centric_inertial_ecliptic,
  planet_centric_star_fixed_equatorial,
  planet_centric_inertial_ecliptic,
  planet_centric_inertial_equatorial,
  planet_centric_star_fixed_ecliptic,
  planet_centric_local );
END_TYPE;

TYPE standard_direction_in_space = ENUMERATION OF (
  gamma_50_vernal_equinox,
  gamma_2000_vernal_equinox,
  TOD_vernal_equinox );
END_TYPE;

ENTITY space_coordinate_system;
  reference_system   : space_coordinate_system_type;
  standard_direction : OPTIONAL standard_direction_in_space;
END_ENTITY;

TYPE orbit_class_type = ENUMERATION OF (
  general,
  geostationary,
  sun_synchronous,
  molniya );
END_TYPE;

ENTITY orbit_arc
  SUBTYPE OF ( space_mission_aspect );
  orbit_class                     : orbit_class_type;
  governing_celestial_body        : celestial_body;
  evaluation_events               : LIST [0:?] OF space_mission_event;
  star_spherical_right_ascension  : OPTIONAL plane_angle_measure;
  star_spherical_declination      : OPTIONAL plane_angle_measure;
  orbit_period                    : OPTIONAL time_measure_with_unit;
END_ENTITY;

ENTITY discretised_orbit_arc
  SUBTYPE OF ( orbit_arc );
  centre_of_governing_celestial_body : cartesian_point;
  positions_and_velocities           : LIST [0:?] OF orbit_position_and_velocity;
  orbit_generator_name               : label;
  kepler_parameters                  : OPTIONAL kepler_parameter_set;
END_ENTITY;

ENTITY keplerian_orbit_arc
  SUBTYPE OF ( orbit_arc );
  kepler_parameters                  : kepler_parameter_set;
END_ENTITY;

ENTITY kepler_parameter_set;
  semi_major_axis                   : length_measure;
  eccentricity                      : REAL;
```

```
  inclination                       : plane_angle_measure;
  right_ascension_of_ascending_node : plane_angle_measure;
  argument_of_periapsis             : plane_angle_measure;
  true_anomaly_at_start             : plane_angle_measure;
WHERE
  wr1: semi_major_axis >= 0.0;
  wr2: eccentricity >= 0.0;
  wr3: (-180.0 < inclination) AND (inclination <= 180.0);
  wr4: (-360.0 < right_ascension_of_ascending_node) AND
       (right_ascension_of_ascending_node <= 360.0);
  wr5: (0.0 <= argument_of_periapsis) AND (argument_of_periapsis < 360.0);
  wr6: (-360.0 < true_anomaly_at_start) AND (true_anomaly_at_start <= 360.0);
END_ENTITY;

ENTITY orbit_position_and_velocity;
  position           : cartesian_point;
  velocity_magnitude : velocity_measure;
  velocity_direction : direction;
END_ENTITY;

TYPE generalized_length = SELECT (
  length_measure,
  function_valued_property );
END_TYPE;

ENTITY celestial_body;
  name                     : label;
  radius                   : generalized_length;
  gravitational_constant   : REAL;
  sidereal_rotation_period : OPTIONAL time_measure;
UNIQUE
  ur1: name;
END_ENTITY;

ENTITY celestial_body_with_orbit
  SUBTYPE OF (celestial_body );
  orbit: orbit_arc;
END_ENTITY;

TYPE generalized_temperature = SELECT (
  thermodynamic_temperature_measure,
  function_valued_property );
END_TYPE;

ENTITY thermal_aspects_of_celestial_body;
  of_celestial_body                   : celestial_body;
  black_body_surface_temperature      : generalized_temperature;
  notional_distance_to_celestial_body : OPTIONAL positive_length_measure;
  spectrum                            : OPTIONAL thermal_radiation_spectrum;
  albedo_reflectors                   : SET [0:?] OF albedo_reflector;
END_ENTITY;

TYPE generalized_albedo_reflection_factor = SELECT (
  ratio_measure,
  function_valued_property );
END_TYPE;

ENTITY albedo_reflector;
  albedo_originator        : thermal_aspects_of_celestial_body;
  albedo_reflection_factor : generalized_albedo_reflection_factor;
  spectrum                 : OPTIONAL thermal_radiation_spectrum;
END_ENTITY;

ENTITY cartesian_translation;
  x_distance : length_measure;
  y_distance : length_measure;
  z_distance : length_measure;
END_ENTITY;
```

```
ENTITY main_body_placement
  SUPERTYPE OF ( ONEOF (
    main_body_placement_with_constant_rotation,
    main_body_placement_with_spinning ) )
  SUBTYPE OF ( space_mission_aspect );
  translation           : OPTIONAL cartesian_translation;
  primary_orientation   : OPTIONAL pointing_in_space;
  secondary_orientation : OPTIONAL pointing_in_space;
  rotations             : LIST [0:?] OF attitude_rotation;
END_ENTITY;

ENTITY main_body_placement_with_constant_rotation
  SUBTYPE OF ( main_body_placement );
  rotation_axis    : axis_placement;
  angular_velocity : angular_velocity_measure;
END_ENTITY;

ENTITY main_body_placement_with_spinning
  SUBTYPE OF ( main_body_placement );
  spin_axis                : axis_placement;
  number_of_spin_positions : OPTIONAL INTEGER;
WHERE
  wr1: number_of_spin_positions > 0;
END_ENTITY;

TYPE named_pointing_in_space = ENUMERATION OF (
  governing_star,
  equatorial_projection_of_governing_star,
  vernal_equinox,
  ecliptic_north_pole,
  planet_centre, planet_zenith,
  planet_north_pole, angular_momentum,
  spacecraft_velocity_vector );
END_TYPE;

TYPE pointing_in_space_select = SELECT (
  named_pointing_in_space,
  general_star_pointing,
  general_space_pointing );
END_TYPE;

ENTITY pointing_in_space;
  pointer          : direction;
  desired_pointing : pointing_in_space_select;
END_ENTITY;

ENTITY general_star_pointing;
  of_case                        : space_mission_case;
  star_name                      : label;
  star_spherical_right_ascension : plane_angle_measure;
  star_spherical_declination     : plane_angle_measure;
UNIQUE
  ur1: of_case, star_name;
WHERE
  wr1: (-360.0 <= star_spherical_right_ascension) AND ( star_spherical_right_
ascension <= 360.0);
  wr2: (-90.0 <= star_spherical_declination) AND ( star_spherical_declination <=
90.0);
END_ENTITY;

ENTITY general_space_pointing;
  of_case   : space_mission_case;
  name      : label;
  direction : direction;
UNIQUE
  ur1: of_case, name;
END_ENTITY;
```

```
TYPE generalized_plane_angle = SELECT (
  plane_angle_measure,
  function_valued_property );
END_TYPE;

ENTITY attitude_rotation;
  axis  : direction;
  angle : generalized_plane_angle;
WHERE
  wr1: (-360.0 <= angle) AND (angle <= 360.0);
END_ENTITY;

ENTITY kinematic_articulation
  SUBTYPE OF ( space_mission_aspect );
  start_pair_value : kinematic_pair_value;
  end_pair_value   : kinematic_pair_value;
END_ENTITY;

ENTITY thermal_environment
  SUBTYPE OF ( space_mission_aspect );
  environmental_sink_name        : label;
  environmental_sink_node_id     : OPTIONAL INTEGER;
  environmental_sink_temperature : thermodynamic_temperature_measure;
  celestial_body_environment     : LIST [0:?] OF thermal_aspects_of_celestial_body;
  star_flux_constant             : OPTIONAL REAL;
END_ENTITY;

ENTITY thermal_radiation_spectrum;
  name        : label;
  description : text;
  values      : function_valued_property;
END_ENTITY;

ENTITY geometric_context;
  length_unit                   : length_unit;
  plane_angle_unit              : plane_angle_unit;
  point_in_plane_uncertainty    : length_measure;
  point_coincidence_uncertainty : length_measure;
END_ENTITY;

TYPE surface_type = SELECT (
  primitive_bounded_surface,
  compound_surface_shell,
  boolean_construction_surface,
  boolean_difference_surface );
END_TYPE;

ENTITY geometric_model;
  id              : identifier;
  name            : label;
  description     : text;
  surfaces        : SET [0:?] OF surface_type;
  submodel_usages : SET [0:?] OF geometric_submodel_usage;
END_ENTITY;

ENTITY geometric_submodel_usage;
  id        : identifier;
  role      : label;
  submodel  : geometric_model;
  placement : axis_placement;
END_ENTITY;

ENTITY geometric_construction_point;
  id          : identifier;
  coordinates : LIST [3:3] OF length_measure;
END_ENTITY;
```

```
ENTITY axis_placement;
  location         : geometric_construction_point;
  z_axis_direction : direction;
  x_axis_direction : direction;
END_ENTITY;

ENTITY primitive_bounded_surface
  SUPERTYPE OF ( ONEOF (
    triangle,
    rectangle,
    quadrilateral,
    disc,
    cylinder,
    cone,
    sphere,
    paraboloid ) );
  id        : identifier;
  placement : OPTIONAL axis_placement;
  colour    : OPTIONAL colour_rgb;
END_ENTITY;

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

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

ENTITY quadrilateral
  SUBTYPE OF ( primitive_bounded_surface );
  p1 : geometric_construction_point;
  p2 : geometric_construction_point;
  p3 : geometric_construction_point;
  p4 : geometric_construction_point;
END_ENTITY;

ENTITY disc
  SUBTYPE OF ( primitive_bounded_surface );
  p1                : geometric_construction_point;
  p2                : geometric_construction_point;
  external_diameter : length_measure;
  internal_diameter : length_measure;
  p3                : geometric_construction_point;
  start_angle       : plane_angle_measure;
  end_angle         : plane_angle_measure;
WHERE
  wr1: (0.0 <= internal_diameter) AND (internal_diameter < external_diameter);
  wr2: (-360.0 < start_angle) AND (start_angle <= 360.0);
  wr3: (-360.0 < end_angle ) AND (end_angle <= 360.0);
  wr4: (0.0 <=  (end_angle-start_angle) ) AND ( (end_angle-start_angle) <= 360.0);
END_ENTITY;

ENTITY cylinder
  SUBTYPE OF ( primitive_bounded_surface );
  p1          : geometric_construction_point;
  p2          : geometric_construction_point;
  diameter    : length_measure;
  p3          : geometric_construction_point;
  start_angle : plane_angle_measure;
  end_angle   : plane_angle_measure;
WHERE
```

```
  wr1: diameter > 0.0;
  wr2: (-360.0 < start_angle) AND (start_angle <= 360.0);
  wr3: (-360.0 < end_angle ) AND (end_angle <= 360.0);
  wr4: (0.0 <=  (end_angle-start_angle) ) AND ( (end_angle-start_angle) <= 360.0);
END_ENTITY;

ENTITY cone
  SUBTYPE OF ( primitive_bounded_surface );
  p1            : geometric_construction_point;
  p2            : geometric_construction_point;
  base_diameter : length_measure;
  apex_diameter : length_measure;
  p3            : geometric_construction_point;
  start_angle   : plane_angle_measure;
  end_angle     : plane_angle_measure;
WHERE
  wr1: base_diameter > 0.0;
  wr2: apex_diameter >= 0.0;
  wr3: (-360.0 < start_angle) AND (start_angle <= 360.0);
  wr4: (-360.0 < end_angle ) AND (end_angle <= 360.0);
  wr5: (0.0 <=  (end_angle-start_angle) ) AND ( (end_angle-start_angle) <= 360.0 );
END_ENTITY;

ENTITY sphere
  SUBTYPE OF ( primitive_bounded_surface );
  p1              : geometric_construction_point;
  p2              : geometric_construction_point;
  diameter        : length_measure;
  base_truncation : length_measure;
  apex_truncation : length_measure;
  p3              : geometric_construction_point;
  start_angle     : plane_angle_measure;
  end_angle       : plane_angle_measure;
WHERE
  wr1: (–diameter/2.0 <= base_truncation) AND
       (base_truncation < apex_truncation) AND
       (apex_truncation <= diameter/2.0);
  wr2: (-360.0 < start_angle) AND (start_angle <= 360.0);
  wr3: (-360.0 < end_angle ) AND (end_angle <= 360.0);
  wr4: (0.0 <=  (end_angle-start_angle) ) AND ( (end_angle-start_angle) <= 360.0);
END_ENTITY;

ENTITY paraboloid
  SUBTYPE OF ( primitive_bounded_surface );
  p1              : geometric_construction_point;
  p2              : geometric_construction_point;
  diameter        : length_measure;
  base_truncation : length_measure;
  apex_truncation : length_measure;
  p3              : geometric_construction_point;
  start_angle     : plane_angle_measure;
  end_angle       : plane_angle_measure;
WHERE
  wr1: diameter > 0.0;
  wr2: (0.0 <= apex_truncation) AND
       (apex_truncation < base_truncation) AND
       (base_truncation <= distance_between(p1,p2) );
  wr3: (-360.0 < start_angle) AND (start_angle <= 360.0);
  wr4: (-360.0 < end_angle ) AND (end_angle <= 360.0);
  wr5: (0.0 <=  (end_angle-start_angle) ) AND ( (end_angle-start_angle) <= 360.0);
END_ENTITY;

TYPE compound_surface_shell_constituent = SELECT (
  primitive_bounded_surface,
  compound_surface_shell );
END_TYPE;

ENTITY compound_surface_shell;
```

```
  id       : identifier;
  name     : label;
  surfaces : LIST [2:?] OF compound_surface_shell_constituent;
END_ENTITY;

TYPE surface_sense_type = ENUMERATION OF (
  positive, negative );
END_TYPE;

ENTITY boolean_construction_surface
  ABSTRACT SUPERTYPE OF ( ONEOF (
    half_space_plane,
    half_space_cylinder ) );
  id        : identifier;
  sense     : surface_sense_type;
  placement : axis_placement;
END_ENTITY;

ENTITY half_space_plane
  SUBTYPE OF ( boolean_construction_surface );
END_ENTITY;

ENTITY half_space_cylinder
  SUBTYPE OF ( boolean_construction_surface );
  radius : length_measure;
END_ENTITY;

ENTITY generalized_half_space;
  base_surface : primitive_bounded_surface;
  sense        : surface_sense_type;
END_ENTITY;

ENTITY boolean_result_surface
  ABSTRACT SUPERTYPE OF ( boolean_difference_surface );
  id        : identifier;
  placement : OPTIONAL axis_placement;
END_ENTITY;

TYPE boolean_first_operand_type = SELECT (
  primitive_bounded_surface,
  compound_bounded_surface,
  boolean_difference_surface,
  geometric_submodel_usage ); (* added per r2.4.1 *)
END_TYPE;

TYPE boolean_second_operand_type = SELECT (
  boolean_construction_surface,
  generalized_half_space );
END_TYPE;

ENTITY boolean_difference_surface
  SUBTYPE OF ( boolean_result_surface );
  first_operand  : boolean_first_operand_type;
  second_operand : boolean_second_operand_type;
END_ENTITY;

ENTITY kinematic_model;
  id          : identifier;
  name        : label;
  description : text;
  joints      : SET [0:?] OF kinematic_joint;
  base        : kinematic_link;
END_ENTITY;

TYPE associated_product_type = SELECT (
  product_definition,
  product_next_higher_assembly_usage_relationship );
END_TYPE;
```

```
ENTITY kinematic_link;
  shape_representation : geometric_model;
  associated_product   : associated_product_type;
END_ENTITY;

ENTITY kinematic_joint;
  first_link  : kinematic_link;
  second_link : kinematic_link;
WHERE
  wr1: first_link :<>: second_link;
END_ENTITY;

ENTITY kinematic_pair
  ABSTRACT SUPERTYPE OF ( ONEOF (
    revolute_kinematic_pair,
    prismatic_kinematic_pair ) );
  id    : identifier;
  joint : kinematic_joint;
END_ENTITY;

ENTITY kinematic_pair_value
  ABSTRACT SUPERTYPE OF ( ONEOF (
    explicit_revolute_kinematic_pair_value,
    implicit_revolute_kinematic_pair_value,
    prismatic_kinematic_pair_value ) );
  applies_to_pair : kinematic_pair;
END_ENTITY;

ENTITY revolute_kinematic_pair
  SUBTYPE OF ( kinematic_pair );
  rotation_axis               : axis_placement;
  lower_limit_actual_rotation : OPTIONAL plane_angle_measure;
  upper_limit_actual_rotation : OPTIONAL plane_angle_measure;
WHERE
  wr1: NOT ( EXISTS(lower_limit_actual_rotation) AND
             EXISTS(upper_limit_actual_rotation) AND
             lower_limit_actual_rotation > upper_limit_actual_rotation );
END_ENTITY;

ENTITY explicit_revolute_kinematic_pair_value
  SUBTYPE OF ( kinematic_pair_value );
  actual_rotation: plane_angle_measure;
END_ENTITY;

ENTITY implicit_revolute_kinematic_pair_value
  SUBTYPE OF ( kinematic_pair_value );
  pointing                          : pointing_in_space;
  may_modify_pointing_of_first_link : BOOLEAN;
END_ENTITY;

ENTITY prismatic_kinematic_pair
  SUBTYPE OF ( kinematic_pair );
  sliding_axis                  : axis_placement;
  lower_limit_actual_translation : OPTIONAL length_measure;
  upper_limit_actual_translation : OPTIONAL length_measure;
WHERE
  wr1: NOT ( EXISTS(lower_limit_actual_translation) AND
             EXISTS(upper_limit_actual_translation) AND
             lower_limit_actual_translation > upper_limit_actual_translation );
END_ENTITY;

ENTITY prismatic_kinematic_pair_value
  SUBTYPE OF ( kinematic_pair_value );
  actual_translation : length_measure;
END_ENTITY;

ENTITY basic_thermal_radiative_model
```

```
  SUBTYPE OF ( network_model );
WHERE
  wr1: QUERY( c <* SELF.constituents |
       'TAS_ARM_R2_4_SCHEMA.THERMAL_RADIATIVE_FACE_SET' IN TYPEOF(c) );
END_ENTITY;

TYPE active_side_type = ENUMERATION OF (
  none,
  positive,
  negative );
END_TYPE;

ENTITY thermal_radiative_face_set
  SUBTYPE OF ( network_node );
  surface            : primitive_bounded_surface;
  active_side        : active_side_type;
  assigned_material  : OPTIONAL material;
  notional_thickness : OPTIONAL length_measure;
END_ENTITY;

ENTITY meshed_thermal_radiative_model
  SUBTYPE OF ( network_model );
WHERE
  wr1: QUERY( c <* SELF.constituents |
       'TAS_ARM_R2_4_SCHEMA.THERMAL_RADIATIVE_MESHING' IN TYPEOF(c) );
END_ENTITY;

TYPE meshing_numbering_option = ENUMERATION OF (
  direction_1_first,
  direction_2_first ) ;
END_TYPE;

ENTITY thermal_radiative_meshing
  SUBTYPE OF ( network_node );
  face_set         : thermal_radiative_face_set;
  meshings_1       : INTEGER;
  meshings_2       : INTEGER;
  numbering_option : meshing_numbering_option;
  grid_spacings    : OPTIONAL LIST [1:?] OF ratio_measure;
END_ENTITY;

ENTITY thermal_radiative_meshing_with_incremental_node_list
  SUBTYPE OF ( thermal_radiative_meshing );
  first_thermal_node_id     : OPTIONAL positive_integer;
  thermal_node_id_increment : INTEGER;
END_ENTITY;

TYPE thermal_node_id_type = INTEGER;
WHERE
  SELF > 0;
END_TYPE;

ENTITY thermal_radiative_meshing_with_full_node_list
  SUBTYPE OF ( thermal_radiative_meshing );
  thermal_node_ids : LIST [1:?] OF thermal_node_id_type;
END_ENTITY;

ENTITY enclosure;
  of_model  : basic_thermal_radiative_model;
  id        : identifier;
  name      : label;
  face_sets : LIST [0:?] OF thermal_radiative_face_set;
UNIQUE
  ur1: of_model, id;
END_ENTITY;

ENTITY colour_rgb;
  red   : ratio_measure;
```

```
  green : ratio_measure;
  blue  : ratio_measure;
WHERE
  wr1: (0 <= red) AND (red <= 1);
  wr2: (0 <= green) AND (green <= 1);
  wr3: (0 <= blue) AND (blue <= 1);
END_ENTITY;

ENTITY material;
  id                : identifier;
  name              : label;
  designation       : OPTIONAL label;
  surface_treatment : OPTIONAL label;
  description       : OPTIONAL text;
UNIQUE
  ur1: id;
END_ENTITY;

ENTITY product_material_relationship;
  of_product       : product_definition;
  bulk_material    : material;
  surface_material : OPTIONAL material;
END_ENTITY;

TYPE generalized_valued_property = SELECT (
  function_valued_property,
  string_valued_property,
  number_valued_property );
END_TYPE;

ENTITY material_property;
  of_material    : material;
  in_environment : material_property_environment;
  property_value : generalized_valued_property;
END_ENTITY;

ENTITY material_property_environment
  SUBTYPE OF ( space_mission_aspect );
END_ENTITY;

ENTITY commercial_aspects_of_material;
  of_material     : material;
  commercial_name : label;
  supplier        : organization;
END_ENTITY;

-- ENTITY derived_unit can be used 'as is' from ISO 10303-41:1994

ENTITY conversion_unit;
  name   : label;
  factor : measure_with_unit;
  offset : measure_with_unit;
END_ENTITY;

ENTITY function_valued_property;
  valued_property : property_class_usage;
  index_value     : INTEGER;
END_ENTITY;

ENTITY string_valued_property;
  valued_property : property_class_usage;
  index_value     : INTEGER;
END_ENTITY;

ENTITY number_valued_property;
  valued_property : property_class_usage;
  number_value    : NUMBER;
END_ENTITY;
```

```
END_SCHEMA;
```

G.2 EXPRESS-G Diagrams of the ARM

NOTE: Only EXPRESS-G diagrams of the most important parts of the ARM are presented.

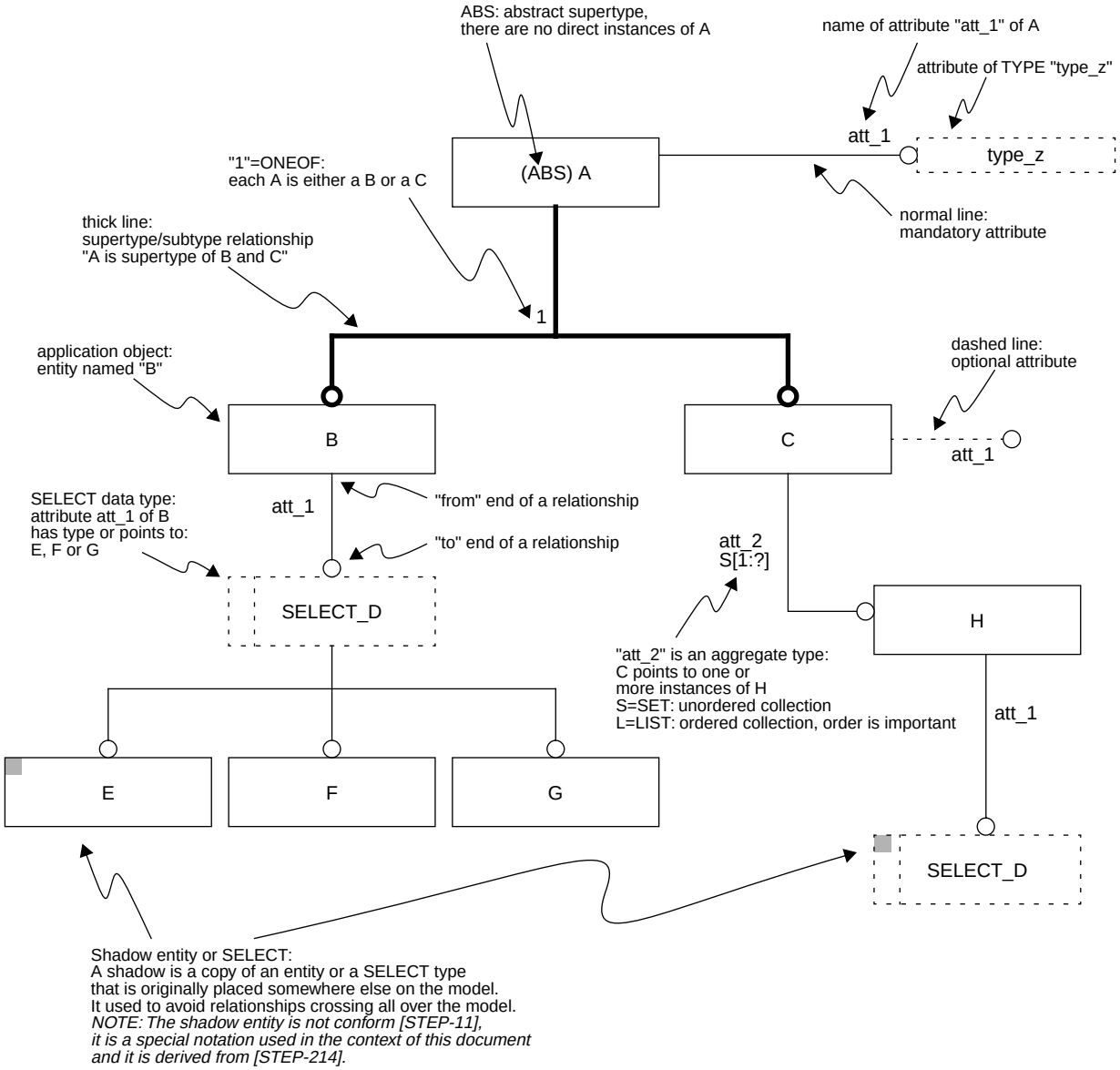


Figure G–1 - Brief overview of the EXPRESS-G graphical notation – adapted from a diagram in [STEP-214]

NOTE – The ARM EXPRESS-G diagrams can be supplied as a separate booklet, automatically generated from the TAS ARM EXPRESS SCHEMA.

Annex H - AIM EXPRESS-G (Informative)

The following diagrams correspond to the AIM EXPRESS short form defined in clause 5. The diagram use the EXPRESS-G graphical notation for the EXPRESS language.

EXPRESS-G is defined in annex D of ISO 10303-11.

NOTE - The EXPRESS-G diagrams can be supplied as a separate booklet generated automatically from the TAS AIM SCHEMA as specified in Annex A.

Annex J - AIM EXPRESS Listing (Informative)

To be supplied as file TAS_R2_4.DOS on a MS-DOS formatted 1.44 MB DS/HD floppy disk.

Annex K - Application protocol implementation and usage guide (Informative)

K.1 Pseudo-algorithm detailing the positioning and orientation of objects in space

The pseudo-algorithm below clarifies how the final position and orientation of a geometrical object – a (sub)model, a surface, a thermal-radiative face – are defined. The final actual position and orientation of a thermal-radiative face at a particular time during an analysis or operation run depend on a sequence of steps established by a hierarchical series of definitions of coordinate system transformations.

1) The basic coordinate system is defined in the *coordinate_system* of the **space_mission_case** referenced in the *of_case* of a **space_mission_phase**.
2) The actual orbit position at a particular run time can be determined from the *evaluation_events* and the *positions_and_velocities* or the *kepler_parameters* of the **orbit_arc** .
3) From the orbit position a *translation* is possible for the whole model as specified in the relevant **main_body_placement** which is referred to in the *space_aspects* of a **space_mission_phase**.
4) Then the *primary_orientation* and possibly *secondary_orientation* are applied, possibly followed by a number of **attitude_rotation**s. These are specified in the relevant **main_body_placement** which is referred to in the *space_aspects* of a **space_mission_phase**.
5) Now the actual coordinate system of the main body – i.e. the root **geometric_model** and corresponding root **basic_thermal_radiative_model** – is defined. The *submodels* inside the **geometric_model** are initially positioned according to their *axis_placement* specification in the corresponding **geometric_submodel_usage** objects.
6) The **bounded_surface**s inside the **geometric_model** or its *submodels* are positioned according to the *axis_placement* attribute in the **bounded_surface** instances.
7) Each of the *submodels* – including its tree of lower level *submodels* – of the **geometric_model** can be considered to be a rigid body, which can move with respect to the main body or with respect to its immedeate *supermodel*. For a **submodel_usage** an independent kinematic articulation can be defined through a **kinematic_model** and a **kinematic_articulation** which is a **space_mission_aspect** referenced by a **space_mission_phase** in its*space_aspects*.

Annex M - Bibliography (Informative)

M.1 ISO 10303 and Related Documents

[STEP-AP-Guide] Guidelines for the development and approval of STEP application protocols, version 1.2, M. Palmer, M. Gilbert, ISO TC184/SC4 N512, 5 April 1996.

[STEP-map-Guide] Guidelines for the development of mapping tables, A. Barnard, D. Craig, ISO TC184/SC4 N436, 5 April 1996.

[STEP-AIM-Guide] Guidelines for AIM development, A. Barnard, M. Gilbert, ISO TC184/SC4 N435, 5 April 1996.

[STEP-Sup-Dir] Supplementary directives for the drafting and presentation of ISO 10303, Joan Wellington, Nigel Shaw ISO TC184/SC4 N432, 5 April 1996.

[STEP-1] ISO 10303-1 (IS) Overview and Fundamental Principles

[STEP-22] ISO 10303-22 (CD) Implementation methods: Standard Data Access Interface (SDAI)

[STEP-203] ISO 10303-203 (IS) Application Protocol: Configuration controlled design

[STEP-209] ISO 10303-209 (CDCE) Application Protocol: Design through analysis of composite and metallic structures (Note: Relevance is that this is the first example of integration of [STEP-104])

[STEP-214] ISO 10303-214 (CD) Application Protocol: Core data for automotive mechanical design processes

[STEP-NRF-AP] STEP Application Protocol – Network-model Results Format, NRF-002-AP, Version 1, Release 1.4, 15 November 1996, ESA/ESTEC/YCV, Noordwijk, The Netherlands.

Note: the edition of the documents is given between parentheses, using the following abbreviations:

CD	Committee Draft
CDB	Committee Draft, in Ballot
CDBE	Committee Draft, Ballot Ended
CDC	Committee Draft, for Comment
CDCE	Committee Draft, Comments Ended
DIS	Draft International Standard
IS	International Standard

M.2 General Reference Documents

[ITT] Invitation to Tender “Thermal Neutral Format Based on STEP”, AO/1-2724/93/NL/FG, 23 August 1993

[Proposal] Proposal Thermal Neutral Format Based on STEP FSS-P-93-0074, Issue 1, 25 October 1993, Proposal to ESA by Fokker Space & Systems and EPSILON Ingénierie, in reply to [ITT].

[PSS-05-0] ESA Software Engineering Standards, ESA PSS-05-0, Issue 2, 1991

[YCV-09] YCV Standards and Guidelines for Software Documentation

[ICETAS-URD] ICETAS User Requirements Document,
URD-ICETAS-008, Release 2.2, 18 February 1994.

[ICETAS-SID] ESA ICETAS Standards Document,
SID-ICETAS-009, Release 2.2, 28 January 1994.

[ICETAS-FR] Final Report ICETAS Phase 2,
FR-ICETAS-061, Release 1.1, 19 April 1994.

[ATS-v1] SET - Protocole d'Application Thermique Spatiale,
GDT-N/92-001, Version 1, Association GOSET, Nanterre, France, 1992.

[ATS-v1-STEP] Spacecraft Thermal Application Protocol using STEP Technology,
GDT-N/93-002, Version 1, 16-Dec-93 (by Association GOSET).

[ATS-v2-URD] SET-ATS Version 2 User Requirements
RE/0294/CP/CNES, February 1994 (by EPSILON Ingénierie)

[ATS-v2] SET Application Protocol - Data for Spatial Thermal Analyses,
GDT-N/93-002, Version 2.1, Association GOSET, Nanterre, France,
14 April 1994.

[Spacecraft-TCDD] Spacecraft Thermal Control Design Data
ESA PSS-03-108 Issue 1, November 1989, ESA/ESTEC,
Noordwijk, The Netherlands

[ESARAD-UM] ESARAD User Manual,
UM-ESARAD-024, Version 3.1, 8 July 1993,
ESTEC/YCV & EGT Engineering Research Centre

[ESARAD-RM] Reference Manual ESARAD Version 2.2C,
UM-ESARAD-083, Version 4.1, 20 July 1994,
ESTEC/YCV & EGT Engineering Research Centre

[THERMICA] THERMICA Version 3.0 User's Manual,
S413/NT/43.94, Issue 1, 1994,
Matra Marconi Space

[ESABASE-RM] ESABASE Reference Manual,
ESABASE/GEN-UM-061, Issue 2, December 1993
ESTEC/WM

[ESABASE-THM] ESABASE Thermal Manual,
ESABASE/THM-UM-043, Draft, December 1993
ESTEC/WM

[Larson&Wertz] Wiley J. Larson and James R. Wertz (editors),
Space Mission Analysis and Design, Second Edition,
Space Technology Library,
1992, Microcosm, Inc., Torrance CA
& Kluwer Academic Publishers, Dordrecht/Boston/London