ISO TC 184/SC4/ WG3__ N1353

Date: 2003/07/24

Supersedes ISO TC 184/SC4/WG3__ N ___

ISO/WD 10303-237
Product data representation and exchange: Application protocol: Fluid dynamics data

ABSTRACT:

This provides a draft of the AP for fluid dynamics data.

KEYWORDS: Computational fluid dynamics, Wind tunnel test, Flight test

COMMENTS TO READER:

This is a revision of the document dated 2001/08/31 following comments at the STEP Myrtle Beach meeting in March 2002. The formal modeling uses EXPRESS Edition 2.
The AP is strongly dependant on parts 52, 53 and 110 which are not yet past the CD stage and hence not necessarily stable. Further, we wish to use modules, which are also not yet available. Consequently the AIM is not provided.

Project Leader: Ray Cosner
Address: Boeing, Phantom Works
PO Box 516,
M/S S106–7126
St. Louis, MO 63166

Telephone: +1 (314) 233–6481
Facsimile: +1 (314) 777–1328
E-mail: raymond.r.cosner@boeing.com

Project Editor: Peter Wilson
Address: Boeing Commercial Airplane
PO Box 3707, M/S 2R–97
Seattle, WA 98124-2207

Telephone: +1 (206) 544-0589
Facsimile: +1 (206) 544-5889
E-mail: peter.r.wilson@boeing.com

ISO copyright office
Case postale 56. CH-1211 Geneva 20
Tel. +41 22 749 01 11
Fax +41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Contents

Page

1 Scope . . . 1

2 Normative references . . . 3

3 Terms, definitions, abbreviations, and symbols . . . 4
3.1 Terms defined in ISO 10303-1 . . . 4
3.2 Other terms and definitions . . . 4
3.3 Abbreviations . . . 4
3.4 Symbols . . . 4

4 Information requirements . . . 7
4.1 Units of functionality . . . 7
4.1.1 management_arm . . . 7
4.1.2 analysis_arm . . . 9
4.2 Application objects . . . 12
4.3 Application assertions . . . 54

5 Application interpreted model . . . 55
5.1 Mapping specification . . . 55
5.2 AIM EXPRESS short listing . . . 59

6 Conformance requirements . . . 60

Annex A (normative) AIM EXPRESS expanded listing . . . 61

Annex B (normative) AIM short names . . . 62

Annex C (normative) Implementation method specific requirements . . . 63

Annex D (normative) Protocol Implementation Conformance Statement (PICS) proforma 64

Annex E (normative) Information object registration . . . 65
E.1 Document identification . . . 65
E.2 Schema identification . . . 65

Annex F (informative) Application activity model . . . 66
F.1 Introduction . . . 66
F.2 Application activity model background . . . 66
F.3 Application activity model definitions . . . 68
F.4 Application Process Description . . . 72
F.5 Process Variant Approaches . . . 80

Annex G (informative) Application reference model . . . 82
G.1 Introduction . . . 82
G.2 Management arm . . . 82
G.2.1 Introduction . . . 83
G.2.2 Fundamental concepts and assumptions . . . 83
G.2.3 management_arm type definitions . . . 84
G.2.3.1 design_or_analysis . . . 84
G.2.3.2 retention_data_select . . . 85
G.2.4 management_arm entity definitions . . . 85
G.2.4.1 date . . . 85
G.2.4.2 identifier . . . 85

G.2.4.3 file . . . 86
G.2.4.4 label . . . 86
G.2.4.5 mass_measure . . . 86
G.2.4.6 text . . . 86
G.2.4.7 work_order . . . 88
G.2.4.8 change_order . . . 88
G.2.4.9 start_order . . . 89
G.2.4.10 work_request . . . 89
G.2.4.11 change_request . . . 90
G.2.4.12 start_request . . . 90
G.2.4.13 assembly . . . 91
G.2.4.14 next_higher_assembly . . . 93
G.2.4.15 promissory_usage . . . 93
G.2.4.16 component_assembly_position . . . 93
G.2.4.17 substitute_part . . . 94
G.2.4.18 supplied_part_version . . . 94
G.2.4.19 supplier . . . 95
G.2.4.20 make_from . . . 95
G.2.4.21 approval . . . 96
G.2.4.22 person_organization . . . 97
G.2.4.23 effectivity . . . 97
G.2.4.24 sequence_effectivity . . . 98
G.2.4.25 date_effectivity . . . 99
G.2.4.26 lot_effectivity . . . 100
G.2.4.27 retention_period . . . 100
G.2.4.28 product_configuration . . . 101
G.2.4.29 product_model . . . 102
G.2.4.30 part . . . 104
G.2.4.31 alternate_part . . . 104
G.2.4.32 part_version . . . 105
G.2.4.33 analysis_version . . . 105
G.2.4.34 analysis_design_version_relationship . . . 106
G.2.4.35 design_discipline_product_definition . . . 107
G.2.4.36 analysis_discipline_product_definition . . . 107
G.2.4.37 analysis . . . 108
G.2.4.38 geometry_element . . . 108
G.2.4.39 geometric_model_representation . . . 110
G.2.4.40 advanced_boundary_representation . . . 111
G.2.4.41 faceted_boundary_representation . . . 111
G.2.4.42 manifold_surface_with_topology . . . 111
G.2.4.43 non_topological_surface_and_wireframe . . . 111
G.2.4.44 point_model . . . 112
G.2.4.45 wireframe_with_topology . . . 112
G.2.4.46 shape_aspect . . . 112
G.2.4.47 shape . . . 113
G.2.4.48 nominal_design_shape . . . 113
G.2.4.49 idealized_analysis_shape . . . 114
G.2.4.50 analysis_shape . . . 114
G.2.4.51 node_shape . . . 114
G.2.4.52 additional_design_information . . . 115
G.2.4.53 specification . . . 116
G.2.4.54 design_specification . . . 116

G.2.4.55 surface_finish_specification . . . 116
G.2.4.56 process_specification . . . 117
G.3 Analysis arm . . . 117
G.3.1 Introduction . . . 118
G.3.2 Fundamental concepts and assumptions . . . 119
G.3.2.1 Domain . . . 119
G.3.2.2 Boundary-conditions . . . 122
G.3.2.3 Equations . . . 124
G.3.2.4 Results . . . 124
G.3.2.5 Numerical data . . . 124
G.3.3 analysis_arm type definitions . . . 135
G.3.3.1 indices_group . . . 135
G.3.3.2 mesh_location . . . 135
G.3.3.3 data_name . . . 139
G.3.3.4 defined_data_name . . . 140
G.3.3.5 cfd_data_name . . . 141
G.3.3.6 nondimensional_parameter_name . . . 142
G.3.3.7 data_class . . . 143
G.3.3.8 defined_data_class . . . 144
G.3.3.9 coordinate_data_name . . . 144
G.3.3.10 Riemann_1D_data_name . . . 146
G.3.3.11 bc_type . . . 147
G.3.3.12 bc_type_simple . . . 147
G.3.3.13 bc_type_compound . . . 150
G.3.3.14 ijk_minmax . . . 150
G.3.3.15 force_moment_data_name . . . 151
G.3.3.16 governing_equation_type . . . 153
G.3.3.17 gas_model_type . . . 154
G.3.3.18 gas_model_data_name . . . 155
G.3.3.19 viscosity_model_type . . . 155
G.3.3.20 viscosity_model_data_name . . . 156
G.3.3.21 thermal_conductivity_model_type . . . 157
G.3.3.22 thermal_conductivity_model_data_name . . . 158
G.3.3.23 turbulence_closure_type . . . 159
G.3.3.24 turbulence_closure_data_name . . . 159
G.3.3.25 turbulence_model_type . . . 160
G.3.3.26 turbulence_model_data_name . . . 161
G.3.3.27 flow_solution_data_name . . . 162
G.3.4 analysis_arm entity definitions . . . 165
G.3.4.1 analysis_model . . . 165
G.3.4.2 analysis_step . . . 167
G.3.4.3 zone . . . 168
G.3.4.4 structured_zone . . . 169
G.3.4.5 unstructured_zone . . . 169
G.3.4.6 mesh . . . 170
G.3.4.7 structured_mesh . . . 170
G.3.4.8 rind . . . 171
G.3.4.9 unstructured_mesh . . . 172
G.3.4.10 vertex_defined_cell . . . 172
G.3.4.11 indices_list . . . 173
G.3.4.12 indices_range . . . 173
G.3.4.13 grid_coordinates . . . 173

G.3.4.14 multiple_mesh_block . . . 174
G.3.4.15 mesh_connectivity . . . 175
G.3.4.16 matched_mesh_connection . . . 175
G.3.4.17 mismatched_mesh_connection . . . 177
G.3.4.18 mismatched_mesh_region . . . 178
G.3.4.19 mesh_abutting . . . 178
G.3.4.20 mesh_overset . . . 179
G.3.4.21 mesh_overset_hole . . . 179
G.3.4.22 mismatched_donor_mesh . . . 179
G.3.4.23 structured_donor_mesh . . . 180
G.3.4.24 unstructured_donor_mesh . . . 180
G.3.4.25 condition . . . 181
G.3.4.26 zone_bc . . . 181
G.3.4.27 boundary_condition . . . 182
G.3.4.28 elements_bc . . . 183
G.3.4.29 element_list_bc . . . 184
G.3.4.30 element_range_bc . . . 184
G.3.4.31 point_list_bc . . . 185
G.3.4.32 point_range_bc . . . 185
G.3.4.33 bc_dataset . . . 186
G.3.4.34 Dirichlet_bc_dataset . . . 188
G.3.4.35 Neumann_bc_dataset . . . 189
G.3.4.36 bc_data . . . 189
G.3.4.37 reference_state . . . 190
G.3.4.38 integral_data . . . 191
G.3.4.39 family . . . 191
G.3.4.40 equation . . . 191
G.3.4.41 equation_set . . . 192
G.3.4.42 governing_equation . . . 193
G.3.4.43 diffusion_equation . . . 193
G.3.4.44 behaviour_model . . . 193
G.3.4.45 diffusion_model . . . 194
G.3.4.46 gas_model . . . 195
G.3.4.47 thermal_conductivity_model . . . 195
G.3.4.48 turbulence_closure . . . 196
G.3.4.49 turbulence_model . . . 196
G.3.4.50 viscosity_model . . . 196
G.3.4.51 result . . . 197
G.3.4.52 solution . . . 197
G.3.4.53 solution_with_rind . . . 198
G.3.4.54 convergence_history . . . 198
G.3.4.55 discrete_data . . . 199
G.3.4.56 discrete_data_with_rind . . . 199
G.3.4.57 data_conversion . . . 200
G.3.4.58 dimensional_units . . . 200
G.3.4.59 dimensional_exponents . . . 200
G.3.4.60 data_array . . . 201
G.3.4.61 geometry_reference . . . 205
G.3.4.62 info . . . 206
G.3.4.63 external . . . 206
G.3.5 analysis_arm function definitions . . . 206
G.3.5.1 multi_array_size . . . 206

Annex H (informative) AIM EXPRESS-G 208

Annex J (informative) AIM EXPRESS listing 209

Bibliography 210

Index 211

Figures

Figure 1 — AP237 planning information model xi
Figure F.1 — A0 Design, build, test and ship product 73
Figure F.2 — A0 Design, build, test and ship product 74
Figure F.3 — A1 Design product 75
Figure F.4 — A12 Conduct analysis of design 76
Figure F.5 — A121 Build and run analysis model 77
Figure F.6 — A122 Extract results 78
Figure F.7 — Data flow through the CFD process 79
Figure G.1 — Partial schema level ARM diagram (page1 of 1) 82
Figure G.2 — Illustration of the major elements for product data management 83
Figure G.3 — Illustration of the major elements for CFD analysis 84
Figure G.4 — Partial entity level diagram of ARM management_arm schema (page 1 of 12) 85
Figure G.5 — Partial entity level diagram of ARM management_arm schema (page 2 of 12) 87
Figure G.6 — Partial entity level diagram of ARM management_arm schema (page 3 of 12) 91
Figure G.7 — Partial entity level diagram of ARM management_arm schema (page 4 of 12) 92
Figure G.8 — Partial entity level diagram of ARM management_arm schema (page 5 of 12) 96
Figure G.9 — Partial entity level diagram of ARM management_arm schema (page 6 of 12) 98
Figure G.10 — Partial entity level diagram of ARM management_arm schema (page 7 of 12) 99
Figure G.11 — Partial entity level diagram of ARM management_arm schema (page 8 of 12) 102
Figure G.12 — Partial entity level diagram of ARM management_arm schema (page 9 of 12) 103
Figure G.13 — Partial entity level diagram of ARM management_arm schema (page 10 of 12) 109
Figure G.14 — Partial entity level diagram of ARM management_arm schema (page 11 of 12) 110
Figure G.15 — Partial entity level diagram of ARM management_arm schema (page 12 of 12) 115
Figure G.16 — Topologically based analysis hierarchy 118
Figure G.17 — Example convention for a 2-D cell center 120
Figure G.18 — Example mesh with rind vertices 120
Figure G.19 — A 1-to-1 abutting interface 121
Figure G.20 — A mismatched abutting interface 121
Figure G.21 — An overset interface 122
Figure G.22 — Hierarchy for boundary-condition structures 123
Figure G.23 — Partial entity level diagram of ARM analysis_arm schema (page 1 of 28) . 126
Figure G.24 — Partial entity level diagram of ARM analysis_arm schema (page 2 of 28) . 126
Figure G.25 — Partial entity level diagram of ARM analysis_arm schema (page 3 of 28) . 127
Figure G.26 — Partial entity level diagram of ARM analysis_arm schema (page 4 of 28) . 127
Figure G.27 — Partial entity level diagram of ARM analysis_arm schema (page 5 of 28) . 128
Figure G.28 — Partial entity level diagram of ARM analysis_arm schema (page 6 of 28) . 128
Figure G.29 — Partial entity level diagram of ARM analysis_arm schema (page 7 of 28) . 129

Figure G.30 — Partial entity level diagram of ARM analysis_arm schema (page 8 of 28) . 130
Figure G.31 — Partial entity level diagram of ARM analysis_arm schema (page 9 of 28) . 131
Figure G.32 — Partial entity level diagram of ARM analysis_arm schema (page 10 of 28) 132
Figure G.33 — Partial entity level diagram of ARM analysis_arm schema (page 11 of 28) 132
Figure G.34 — Partial entity level diagram of ARM analysis_arm schema (page 12 of 28) 133
Figure G.35 — Partial entity level diagram of ARM analysis_arm schema (page 13 of 28) 133
Figure G.36 — Partial entity level diagram of ARM analysis_arm schema (page 14 of 28) 134
Figure G.37 — Partial entity level diagram of ARM analysis_arm schema (page 15 of 28) 134
Figure G.38 — Partial entity level diagram of ARM analysis_arm schema (page 16 of 28) 134
Figure G.39 — Partial entity level diagram of ARM analysis_arm schema (page 17 of 28) 134
Figure G.40 — Partial entity level diagram of ARM analysis_arm schema (page 18 of 28) 135
Figure G.41 — Partial entity level diagram of ARM analysis_arm schema (page 19 of 28) 135
Figure G.42 — Partial entity level diagram of ARM analysis_arm schema (page 20 of 28) 136
Figure G.43 — Partial entity level diagram of ARM analysis_arm schema (page 21 of 28) 137
Figure G.44 — Partial entity level diagram of ARM analysis_arm schema (page 22 of 28) 137
Figure G.45 — Partial entity level diagram of ARM analysis_arm schema (page 23 of 28) 138
Figure G.46 — Partial entity level diagram of ARM analysis_arm schema (page 24 of 28) 138
Figure G.47 — Partial entity level diagram of ARM analysis_arm schema (page 25 of 28) 138
Figure G.48 — Partial entity level diagram of ARM analysis_arm schema (page 26 of 28) 139
Figure G.49 — Partial entity level diagram of ARM analysis_arm schema (page 27 of 28) 140
Figure G.50 — Partial entity level diagram of ARM analysis_arm schema (page 28 of 28) 141
Figure G.51 — Parametric coordinate system for a 3-D structured mesh 171

Tables

Table 1 — Symbols for dimensional units 5
Table 2 — Symbols for coordinate systems 5
Table 3 — Symbols for unit vectors 5
Table 4 — Symbols for physical properties 5
Table 5 — Symbols for nondimensional parameters and related scales 6
Table G.1 — Nondimensional parameter name identifiers 143
Table G.2 — Coordinate data name identifiers 145
Table G.3 — Riemann 1-D data name identifiers 147
Table G.4 — Force and moment data name identifiers 152
Table G.5 — Thermal conductivity model data name identifiers 158
Table G.6 — Turbulence data name identifiers 161
Table G.7 — Flow solution data name identifiers 163
Table G.8 — inward_normal_index values 183
Table G.9 — Associated boundary-condition types and usage rules 187
Table G.10 — Data that may be associated with **reference_state** 190
Table G.11 — Encoding of the 3-D **diffusion_model** terms 194

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 10303–237 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC4, *Industrial data.*

This International Standard is organized as a series of parts, each published separately. The structure of this International Standard is described in ISO 10303–1.

Each part of this International Standard is a member of one of the following series: description methods, implementation methods, conformance testing methodology and framework, integrated generic resources, integrated application resources, application protocols, abstract test suites, application interpreted constructs, and application modules. This part is a member of the application protocol series.

A complete list of parts of ISO 10303 is available from the Internet:

`<http://www.nist.gov/sc4/editing/step/titles/>`

Annexes A, B, C, D, and E are a normative part of this part of ISO 10303. Annexes G, H, and J are for information only.

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation of product information and for the exchange of product data. The objective is to provide a neutral mechanism capable of describing products throughout their life cycle. This mechanism is suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and as a basis for archiving.

This part of ISO 10303 specifies an application protocol (AP) for the exchange of computer interpretable computational fluid dynamics (CFD) information, including product shape, associated meshes defining the computational domain and CFD boundary conditions, equations and computational results.

The shape definitions for design and analysis are each independently configuration controlled.

A high level planning information model for this application protocol is shown in Figure 1. At this level the product can be conceptualized as a part that has both design and analysis product definitions. Each definition has one or more shape representations. The analysis product definition has associated CFD analysis models, boundary conditions, equations and analysis results in addition to its shape representations.

The two possible shape representations in this AP include the nominal design shape and an idealized analysis shape. The nominal design shape includes geometry and topology for the part. The idealized analysis shape need include only the geometry and topology for the part/fluid boundary.

This part of ISO 10303 satisfies the need for exchange of information between the iterative design and CFD analysis stages of the product life cycle.

Application protocols provide the basis for developing implementations of ISO 10303 and abstract test suites for the conformance testing of AP implementations.

Clause 1 defines the scope of the application protocol and summarizes the functionality and data covered by the AP. Clause 3 lists the words defined in this part of ISO 10303 and gives pointers to words defined elsewhere. An application activity model that is the basis for the definition of the scope is provided in annex F. The information requirements of the application are specified in clause 4 using terminology appropriate to the application. A graphical representation of the information requirements, referred to as the application reference model, is given in annex G.

Resource constructs are interpreted to meet the information requirements. This interpretation produces the application interpreted model (AIM). This interpretation, given in 5.1, shows the correspondence between the information requirements and the AIM. The short listing of the AIM specifies the interface to the integrated resources and is given in 5.2. Note that the definitions and EXPRESS 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. The expanded listing given in annex A contains the complete EXPRESS for the AIM without annotation. A graphical representation of the AIM is given in annex H. Additional requirements for specific implementation methods are given in annex C.

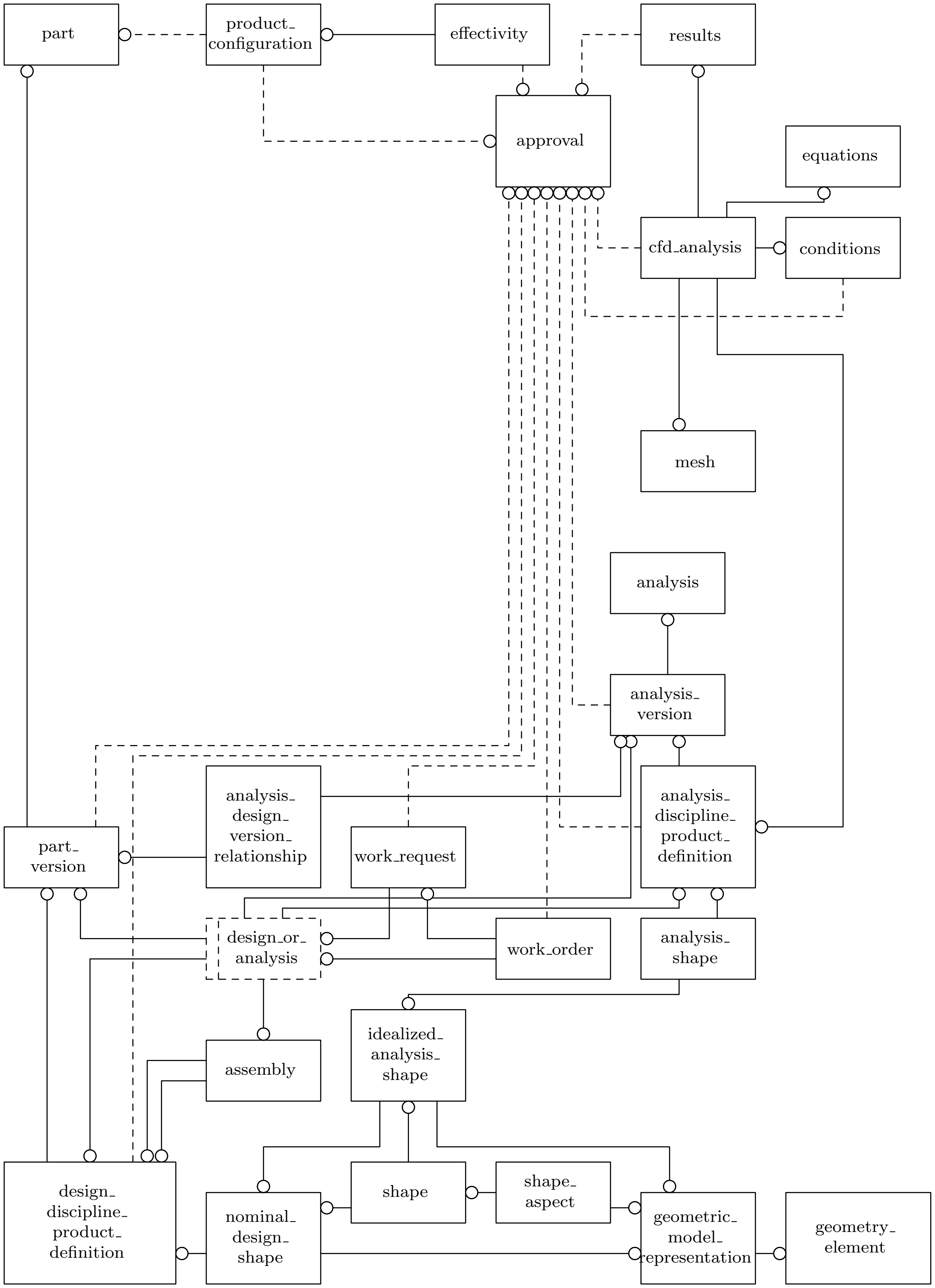

Figure 1 – AP237 planning information model

Industrial automation systems and integration — Product data representation and exchange — Part 237 : Application protocol: Fluid dynamics data

1 Scope

This part of ISO 10303 specifies the use of the integrated resources necessary for the scope and information requirements for the explicit representation of computer-readable data in the discipline of fluid dynamics. This standard will be applicable to all industries requiring representation of a fluid dynamic flowfield, including aerospace, automotive, and shipbuilding industries.

NOTE The application activity model in annex F provides a graphical representation of the processes and information flows that are the basis for the definition of the scope of this part of ISO 10303.

The following are within the scope of this part of ISO 10303:

— digital data on structured and unstructured grids describing steady or unsteady fluid dynamics flowfields;

— data describing the fluid dynamics model including grid description, grid inter-connectivity, boundary conditions, and modeling parameters;

— data from solutions of equation sets commonly used in fluid dynamics analysis: Navier-Stokes equations, Euler equations, linear and nonlinear potential flow equations, small-disturbance equations, boundary layer equations, and stream function equations;

— single-phase flow of a liquid or a gas;

— laminar flow, transitional flow, turbulent flow (direct representation of turbulence, or represented by Reynolds-averaged data);

— incompressible or compressible flow;

— unsteady flow;

— perfect gas, or variable chemical composition (equilibrium flow, frozen flow, or finite-rate chemical reactions);

— data regarding the exchange of energy by molecular transport including convection, conduction, and advectation;

— rotating flowfields (e.g., turbomachinery);

— inertial and rotating frames of reference;

- Newtonian transport laws;
- reference to product geometry;
- administrative information necessary to track the approval and configuration control of the analysis of a product;
- transfer of partial fluid dynamics data;

The following are outside the scope of this part of ISO 10303:

- representations of geometry;
- gross flow in networks (e.g., piping and ducting);
- the use that application programs may make of the data;
- the means by which application programs modify the data;
- the form in which the data is stored internal to an application.

The validity, accuracy and completeness of the data for a particular purpose are determined entirely by the applications' software.

NOTE The following are outside the scope of this edition of this part of ISO 10303 but are expected to be inside the scopes of later editions of this part:

- two- and three-phase flow;
- free surface flow;
- non-continuum flow (e.g., direct simulation of Monte Carlo data);
- data from non-analytical sources (e.g., experimental simulation such as wind tunnel or water tank testing, and product test such as flight test or sea trials);
- data regarding the exchange of energy by radiation;
- non-Newtonian transport laws;
- electro-magnetic interactions with a fluid;
- plasmas.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this international standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this international standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 10303–1:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.*

ISO 10303–11:2003, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description method: The EXPRESS language reference manual.*

ISO 10303–21:2002, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation method: Clear text encoding of the exchange structure.*

ISO 10303–22:1998, *Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation method: Standard data access interface.*

ISO 10303–41:2000, *Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support.*

ISO 10303–42:2003, *Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resource: Geometric and topological representation.*

ISO 10303–43:2000, *Industrial automation systems and integration — Product data representation and exchange — Part 43: Integrated generic resource: Representation structures.*

ISO 10303–50:2002, *Industrial automation systems and integration — Product data representation and exchange — Part 50: Integrated generic resource: Mathematical constructs.*

ISO 10303–51:—[1], *Industrial automation systems and integration — Product data representation and exchange — Part 51: Integrated generic resource: Mathematical description.*

ISO 10303–52:—[1], *Industrial automation systems and integration — Product data representation and exchange — Part 52: Integrated generic resource: Mesh-based topology.*

ISO 10303–53:—[1], *Industrial automation systems and integration — Product data representation and exchange — Part 53: Integrated generic resource: Numerical analysis.*

ISO 10303–104:2000, *Industrial automation systems and integration — Product data representation and exchange — Part 104: Integrated application resource: Finite element analysis.*

ISO 10303–110:—[1], *Industrial automation systems and integration — Product data representation and exchange — Part 110: Integrated application resource: Mesh-based computational fluid dynamics.*

[1] To be published.

ISO/IEC 8824-1:1998, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

3 Terms, definitions, abbreviations, and symbols

3.1 Terms defined in ISO 10303-1

— application protocol (AP)

— integrated resource (IR)

3.2 Other terms and definitions

3.2.1
BC patch
the subrange of a face of a zone where a given boundary-condition is applied

3.2.2
computational fluid dynamics
the set of knowledge and tools used to generate exact or approximate solutions to the mathematical equations governing the motion of a fluid (gas or liquid).

NOTE 1 The underlying knowledge is implemented in computing codes or application programs.

3.2.3
global BC data
boundary-condition data applied globally to a BC patch; for example, specifying a uniform total pressure at an inflow boundary

3.2.4
local BC data
boundary-condition data applied at each grid point of a BC patch; an example of this is varying total pressure specified at each vertex of a BC patch

3.3 Abbreviations

CFD	computational fluid dynamics

3.4 Symbols

Symbols for dimensional units are given in Table 1.

EXAMPLE 1 A length has dimensions $\mathbf{L}$, an area has dimensions $\mathbf{L}^2$, and a velocity has dimensions $\mathbf{L}/\mathbf{T}$ (alternatively written as $\mathbf{LT}^{-1}$).

Symbols for coordinate systems are given in Table 2.

Associated with the coordinate systems are unit vectors, the symbols for which are given in Table 3.

Symbols for physical properties are given in Table 4.

Table 1 – Symbols for dimensional units

Symbol	Description
M	mass unit
L	length unit
T	time unit
Θ	temperature unit
α	angle unit

Table 2 – Symbols for coordinate systems

Symbol	Description
x, y, z	coordinates in a Cartesian system
r, θ, z	coordinates in a Cylindrical system
r, θ, ϕ	coordinates in a Spherical system
ξ, η, ζ	coordinates in an auxiliary system

Table 3 – Symbols for unit vectors

Symbol	Direction	Symbol	Direction	Symbol	Direction
$\hat{e}_x$	x-direction	$\hat{e}_r$	r-direction	$\hat{e}_\xi$	ξ-direction
$\hat{e}_y$	y-direction	$\hat{e}_\theta$	θ-direction	$\hat{e}_\eta$	η-direction
$\hat{e}_z$	z-direction	$\hat{e}_\phi$	ϕ-direction	$\hat{e}_\zeta$	ζ-direction

Table 4 – Symbols for physical properties

Symbol	Description
ρ	static density
p	static pressure
T	static temperature
e	static internal energy per unit mass
h	static enthalpy per unit mass
s	entropy
ρ_0	stagnation density
p_0	stagnation pressure
T_0	stagnation temperature
e_0	stagnation energy per unit mass
h_0	stagnation enthalpy per unit mass
ρe_0	stagnation energy per unit volume
ν	kinematic viscosity ($\nu = \mu/\rho$)
μ	molecular viscosity
ν_t	eddy viscosity
k	thermal conductivity coefficient
R	ideal gas constant ($R = c_p - c_v$)
c_p	specific heat at constant pressure
c_v	specific heat at constant volume

Table 5 – Symbols for nondimensional parameters and related scales

Symbol	Description
M	Mach number ($M = q/c$)
q	Mach velocity scale
c	Mach speed of sound scale
R_e	Reynolds number ($R_e = VL/\nu$)
V	Reynolds velocity scale
L	Reynolds length scale
ν	Reynolds kinematic viscosity scale
Pr	Prandtl number ($Pr = \mu c_p/k$)
k	Prandtl thermal conductivity scale
μ	Prandtl molecular viscosity scale
c_p	Prandtl specific heat scale
γ	specific heat ratio ($\gamma = c_p/c_v$)
c_p	specific heat at constant pressure
c_v	specific heat at constant volume

Symbols for nondimensional parameters are given in Table 5

4 Information requirements

This clause specifies the information required for the exchange of fluid dynamics data.

The information requirements are specified as a set of units of functionality, application objects, and application assertions. These assertions pertain to individual application objects and to relationships between application objects. The information requirements are defined using the terminology of the subject area of this application protocol.

NOTE 1 A graphical representation of the information requirements is given in annex G.

NOTE 2 The information requirements correspond to those of the activities identified as being within the scope of this application protocol in annex F.

NOTE 3 The mapping specification specified in 5.1 shows how the integrated resources are used to meet the information requirements of this application protocol.

4.1 Units of functionality

This subclause specifies the units of functionality for the fluid dynamics data application protocol. This part of ISO 10303 specifies the following units of functionality:

— management_arm;

— analysis_arm.

The units of functionality and a description of the functions that each UoF supports are given below. The application objects included in the UoFs are defined in 4.2.

4.1.1 management_arm

The management_arm UoF contains the information necessary to . . .

The following application objects are used by the management_arm UoF:

— Additional_design_information;

— Advanced_boundary_representation;

— Alternate_part;

— Analysis;

— Analysis_design_version_relationship;

— Analysis_discipline_product_definition;

— Analysis_shape;

— Analysis_version;

— Approval;

- Assembly;
- Change_order;
- Change_request;
- Component_assembly_position;
- Date;
- Date_effectivity;
- Design_discipline_product_definition;
- Design_specification;
- Effectivity;
- Faceted_boundary_representation;
- File;
- Geometric_model_representation;
- Geometry_element;
- Idealized_analysis_shape;
- Identifier;
- Label;
- Lot_effectivity;
- Make_from;
- Manifold_surface_with_topology;
- Mass_measure;
- Next_higher_assembly;
- Node_shape;
- Nominal_design_shape;
- Non_topological_surface_and_wireframe;
- Part;
- Part_version;
- Person_organization;

- Point_model;
- Process_specification;
- Product_configuration;
- Product_model;
- Promissory_usage;
- Retention_period;
- Sequence_effectivity;
- Shape;
- Shape_aspect;
- Specification;
- Start_order;
- Start_request;
- Substitute_part;
- Supplied_part_version;
- Supplier;
- Surface_finish_specification;
- Text;
- Wireframe_with_topology;
- Work_order;
- Work_request.

4.1.2 analysis_arm

The analysis_arm UoF contains the information necessary to . . .

The following application objects are used by the analysis_arm UoF:

- Analysis_model;
- Analysis_step;
- Bc_data;
- Bc_dataset;

- Behaviour_model;
- Boundary_condition;
- Condition;
- Convergence_history;
- Data_array;
- Data_conversion;
- Diffusion_equation;
- Diffusion_model;
- Dimensional_exponents;
- Dimensional_units;
- Dirichlet_bc_dataset;
- Discrete_data;
- Discrete_data_with_rind;
- Element_list_bc;
- Element_range_bc;
- Elements_bc;
- Equation;
- Equation_set;
- External;
- Family;
- Gas_model;
- Geometry_reference;
- Governing_equation;
- Grid_coordinates;
- Indices_list;
- Indices_range;
- Info;

- Integral_data;
- Matched_mesh_connection;
- Mesh;
- Mesh_abutting;
- Mesh_cell_data;
- Mesh_connectivity;
- Mesh_data;
- Mesh_overset;
- Mesh_overset_hole;
- Mesh_vertex_data;
- Mismatched_donor_mesh;
- Mismatched_mesh_connection;
- Mismatched_mesh_region;
- Multiple_mesh_block;
- Neumann_bc_dataset;
- Point_list_bc;
- Point_range_bc;
- Reference_state;
- Result;
- Rind;
- Solution;
- Solution_with_rind;
- Structured_donor_mesh;
- Structured_mesh;
- Structured_zone;
- Thermal_conductivity_model;
- Turbulence_closure;

- Turbulence_model;
- Unstructured_donor_mesh;
- Unstructured_mesh;
- Unstructured_zone;
- Vertex_defined_cell;
- Viscosity_model;
- Zone;
- Zone_bc.

4.2 Application objects

This subclause specifies the application objects for the fluid dynamics data 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 application objects and their definitions are given below.

4.2.1 Additional_design_information

An Additional_design_information ... The data associated with an Additional_design_information are the following:

- additional_information;
- design.

4.2.1.1 additional_information

The additional_information specifies ...

4.2.1.2 design

The design specifies ...

4.2.2 Advanced_boundary_representation

An Advanced_boundary_representation ...

4.2.3 Alternate_part

An Alternate_part ... The data associated with an Alternate_part are the following:

- alternate.

4.2.3.1 alternate

The alternate specifies ...

4.2.4 Analysis

An Analysis . . . The data associated with an Analysis are the following:

— analysis_type;

— owner.

4.2.4.1 analysis_type

The analysis_type specifies . . .

4.2.4.2 owner

The owner specifies . . .

4.2.5 Analysis_design_version_relationship

An Analysis_design_version_relationship . . . The data associated with an Analysis_design_version_relationship are the following:

— analysis;

— design.

4.2.5.1 analysis

The analysis specifies . . .

4.2.5.2 design

The design specifies . . .

4.2.6 Analysis_discipline_product_definition

An Analysis_discipline_product_definition . . . The data associated with an Analysis_discipline_product_definition are the following:

— approved_by;

— creation_date;

— creator;

— description;

— discipline_identification;

— version.

4.2.6.1 approved_by

The approved_by specifies . . .

4.2.6.2 creation_date

The creation_date specifies ...

4.2.6.3 creator

The creator specifies ...

4.2.6.4 description

The description specifies ...

4.2.6.5 discipline_identification

The discipline_identification specifies ...

4.2.6.6 version

The version specifies ...

4.2.7 Analysis_shape

An Analysis_shape ... The data associated with an Analysis_shape are the following:

— analysis_view.

4.2.7.1 analysis_view

The analysis_view specifies ...

4.2.8 Analysis_version

An Analysis_version ... The data associated with an Analysis_version are the following:

— analysis_number;

— approved_by;

— contract_number;

— creator;

— release_status;

— revision_letter;

— security_code.

4.2.8.1 analysis_number

The analysis_number specifies ...

4.2.8.2 approved_by

The approved_by specifies ...

4.2.8.3 contract_number

The contract_number specifies ...

4.2.8.4 creator

The creator specifies ...

4.2.8.5 release_status

The release_status specifies ...

4.2.8.6 revision_letter

The revision_letter specifies ...

4.2.8.7 security_code

The security_code specifies ...

4.2.9 Approval

An Approval ... The data associated with an Approval are the following:

- authorized_by;
- effective_date;
- purpose;
- status.

4.2.9.1 authorized_by

The authorized_by specifies ...

4.2.9.2 effective_date

The effective_date specifies ...

4.2.9.3 purpose

The purpose specifies ...

4.2.9.4 status

The status specifies ...

4.2.10 Assembly

An Assembly . . . The data associated with an Assembly are the following:

- assembly_part;
- component_part;
- security_code.

4.2.10.1 assembly_part

The assembly_part specifies . . .

4.2.10.2 component_part

The component_part specifies . . .

4.2.10.3 security_code

The security_code specifies . . .

4.2.11 Change_order

A Change_order . . . The data associated with a Change_order are the following:

- adopted_solution;
- change_date.

4.2.11.1 adopted_solution

The adopted_solution specifies . . .

4.2.11.2 change_date

The change_date specifies . . .

4.2.12 Change_request

A Change_request . . . The data associated with a Change_request are the following:

- consequence;
- recommended_solution;
- version.

4.2.12.1 consequence

The consequence specifies . . .

4.2.12.2 recommended_solution

The recommended_solution specifies ...

4.2.12.3 version

The version specifies ...

4.2.13 Component_assembly_position

A Component_assembly_position ... The data associated with a Component_assembly_position are the following:

- assembly_shape;
- component_shape;
- definition;
- transformation.

4.2.13.1 assembly_shape

The assembly_shape specifies ...

4.2.13.2 component_shape

The component_shape specifies ...

4.2.13.3 definition

The definition specifies ...

4.2.13.4 transformation

The transformation specifies ...

4.2.14 Date

A Date ...

4.2.15 Date_effectivity

A Date_effectivity ... The data associated with a Date_effectivity are the following:

- end_date;
- start_date.

4.2.15.1 end_date

The end_date specifies ...

4.2.15.2 start_date

The start_date specifies . . .

4.2.16 Design_discipline_product_definition

A Design_discipline_product_definition . . . The data associated with a Design_discipline_product_definition are the following:

- approved_by;
- cad_filename;
- creation_date;
- creator;
- description;
- discipline_identification;
- version.

4.2.16.1 approved_by

The approved_by specifies . . .

4.2.16.2 cad_filename

The cad_filename specifies . . .

4.2.16.3 creation_date

The creation_date specifies . . .

4.2.16.4 creator

The creator specifies . . .

4.2.16.5 description

The description specifies . . .

4.2.16.6 discipline_identification

The discipline_identification specifies . . .

4.2.16.7 version

The version specifies . . .

4.2.17 Design_specification

A Design_specification . . .

4.2.18 Effectivity

An Effectivity . . . The data associated with an Effectivity are the following:

— affected_assemblies;

— approved_by;

— configuration_item.

4.2.18.1 affected_assemblies

The affected_assemblies specifies . . .

4.2.18.2 approved_by

The approved_by specifies . . .

4.2.18.3 configuration_item

The configuration_item specifies . . .

4.2.19 Faceted_boundary_representation

A Faceted_boundary_representation . . .

4.2.20 File

A File . . .

4.2.21 Geometric_model_representation

A Geometric_model_representation . . . The data associated with a Geometric_model_representation are the following:

— elements;

— role.

4.2.21.1 elements

The elements specifies . . .

4.2.21.2 role

The role specifies . . .

4.2.22 Geometry_element

A Geometry_element . . .

4.2.23 Idealized_analysis_shape

An Idealized_analysis_shape . . . The data associated with an Idealized_analysis_shape are the following:

— basis;

— defining_shape.

4.2.23.1 basis

The basis specifies . . .

4.2.23.2 defining_shape

The defining_shape specifies . . .

4.2.24 Identifier

An Identifier . . .

4.2.25 Label

A Label . . .

4.2.26 Lot_effectivity

A Lot_effectivity . . . The data associated with a Lot_effectivity are the following:

— lot_number;

— lot_size;

— lot_size_unit_of_measure.

4.2.26.1 lot_number

The lot_number specifies . . .

4.2.26.2 lot_size

The lot_size specifies . . .

4.2.26.3 lot_size_unit_of_measure

The lot_size_unit_of_measure specifies . . .

4.2.27 Make_from

A Make_from ... The data associated with a Make_from are the following:

- basis;
- resultant.

4.2.27.1 basis

The basis specifies ...

4.2.27.2 resultant

The resultant specifies ...

4.2.28 Manifold_surface_with_topology

A Manifold_surface_with_topology ...

4.2.29 Mass_measure

A Mass_measure ...

4.2.30 Next_higher_assembly

A Next_higher_assembly ... The data associated with a Next_higher_assembly are the following:

- as_required;
- component_quantity;
- reference_designator;
- unit_of_measure.

4.2.30.1 as_required

The as_required specifies ...

4.2.30.2 component_quantity

The component_quantity specifies ...

4.2.30.3 reference_designator

The reference_designator specifies ...

4.2.30.4 unit_of_measure

The unit_of_measure specifies ...

4.2.31 Node_shape

A Node_shape . . .

4.2.32 Nominal_design_shape

A Nominal_design_shape . . . The data associated with a Nominal_design_shape are the following:

- defining_shape;
- design_view.

4.2.32.1 defining_shape

The defining_shape specifies . . .

4.2.32.2 design_view

The design_view specifies . . .

4.2.33 Non_topological_surface_and_wireframe

A Non_topological_surface_and_wireframe . . .

4.2.34 Part

A Part . . . The data associated with a Part are the following:

- owner;
- part_classification;
- part_nomenclature;
- part_number;
- part_type;
- standard_part.

4.2.34.1 owner

The owner specifies . . .

4.2.34.2 part_classification

The part_classification specifies . . .

4.2.34.3 part_nomenclature

The part_nomenclature specifies . . .

4.2.34.4 part_number

The part_number specifies . . .

4.2.34.5 part_type

The part_type specifies . . .

4.2.34.6 standard_part

The standard_part specifies . . .

4.2.35 Part_version

A Part_version . . . The data associated with a Part_version are the following:

- approved_by;
- contract_number;
- creator;
- make_or_buy_code;
- part_number;
- release_status;
- revision_letter;
- security_code;
- weight.

4.2.35.1 approved_by

The approved_by specifies . . .

4.2.35.2 contract_number

The contract_number specifies . . .

4.2.35.3 creator

The creator specifies . . .

4.2.35.4 make_or_buy_code

The make_or_buy_code specifies . . .

4.2.35.5 part_number

The part_number specifies . . .

4.2.35.6 release_status

The release_status specifies . . .

4.2.35.7 revision_letter

The revision_letter specifies . . .

4.2.35.8 security_code

The security_code specifies . . .

4.2.35.9 weight

The weight specifies . . .

4.2.36 Person_organization

A Person_organization . . . The data associated with a Person_organization are the following:

- address;
- organization;
- person;
- person_organization_identification.

4.2.36.1 address

The address specifies . . .

4.2.36.2 organization

The organization specifies . . .

4.2.36.3 person

The person specifies . . .

4.2.36.4 person_organization_identification

The person_organization_identification specifies . . .

4.2.37 Point_model

A Point_model . . .

4.2.38 Process_specification

A Process_specification . . .

4.2.39 Product_configuration

A Product_configuration . . . The data associated with a Product_configuration are the following:

- approved_by;
- item_identification;
- model_name;
- parts_configured;
- phase_of_product.

4.2.39.1 approved_by

The approved_by specifies . . .

4.2.39.2 item_identification

The item_identification specifies . . .

4.2.39.3 model_name

The model_name specifies . . .

4.2.39.4 parts_configured

The parts_configured specifies . . .

4.2.39.5 phase_of_product

The phase_of_product specifies . . .

4.2.40 Product_model

A Product_model . . . The data associated with a Product_model are the following:

- model_name.

4.2.40.1 model_name

The model_name specifies . . .

4.2.41 Promissory_usage

A Promissory_usage . . .

4.2.42 Retention_period

A Retention_period . . . The data associated with a Retention_period are the following:

- approved_by;
- earliest_end_definition;
- is_applied_to;
- latest_end_definition;
- retention_purpose;
- start_definition.

4.2.42.1 approved_by

The approved_by specifies ...

4.2.42.2 earliest_end_definition

The earliest_end_definition specifies ...

4.2.42.3 is_applied_to

The is_applied_to specifies ...

4.2.42.4 latest_end_definition

The latest_end_definition specifies ...

4.2.42.5 retention_purpose

The retention_purpose specifies ...

4.2.42.6 start_definition

The start_definition specifies ...

4.2.43 Sequence_effectivity

A Sequence_effectivity ... The data associated with a Sequence_effectivity are the following:

- component_quantity;
- from_effectivity_identification;
- quantity_unit_of_measure;
- to_effectivity_identification.

4.2.43.1 component_quantity

The component_quantity specifies ...

4.2.43.2 from_effectivity_identification

The from_effectivity_identification specifies ...

4.2.43.3 quantity_unit_of_measure

The quantity_unit_of_measure specifies ...

4.2.43.4 to_effectivity_identification

The to_effectivity_identification specifies ...

4.2.44 Shape

A Shape ... The data associated with a Shape are the following:

- role.

4.2.44.1 role

The role specifies ...

4.2.45 Shape_aspect

A Shape_aspect ... The data associated with a Shape_aspect are the following:

- characteristics;
- geometry;
- parent_shape.

4.2.45.1 characteristics

The characteristics specifies ...

4.2.45.2 geometry

The geometry specifies ...

4.2.45.3 parent_shape

The parent_shape specifies ...

4.2.46 Specification

A Specification ... The data associated with a Specification are the following:

- specification_code;
- specification_source.

4.2.46.1 specification_code

The specification_code specifies ...

4.2.46.2 specification_source

The specification_source specifies ...

4.2.47 Start_order

A Start_order ...

4.2.48 Start_request

A Start_request ...

4.2.49 Substitute_part

A Substitute_part ... The data associated with a Substitute_part are the following:

- base;
- substitute.

4.2.49.1 base

The base specifies ...

4.2.49.2 substitute

The substitute specifies ...

4.2.50 Supplied_part_version

A Supplied_part_version ... The data associated with a Supplied_part_version are the following:

- approved_by;
- certification_required;
- is_identified_as;
- produced_by;
- supplier_part_number.

4.2.50.1 approved_by

The approved_by specifies ...

4.2.50.2 certification_required

The certification_required specifies ...

4.2.50.3 is_identified_as

The is_identified_as specifies . . .

4.2.50.4 produced_by

The produced_by specifies . . .

4.2.50.5 supplier_part_number

The supplier_part_number specifies . . .

4.2.51 Supplier

A Supplier . . . The data associated with a Supplier are the following:

- identified_as;
- supplier_identification.

4.2.51.1 identified_as

The identified_as specifies . . .

4.2.51.2 supplier_identification

The supplier_identification specifies . . .

4.2.52 Surface_finish_specification

A Surface_finish_specification . . .

4.2.53 Text

A Text . . .

4.2.54 Wireframe_with_topology

A Wireframe_with_topology . . .

4.2.55 Work_order

A Work_order . . . The data associated with a Work_order are the following:

- additional_data;
- analysis_data;
- approved_by;
- incorporates;

- versions;
- work_order_identification.

4.2.55.1 additional_data

The additional_data specifies ...

4.2.55.2 analysis_data

The analysis_data specifies ...

4.2.55.3 approved_by

The approved_by specifies ...

4.2.55.4 incorporates

The incorporates specifies ...

4.2.55.5 versions

The versions specifies ...

4.2.55.6 work_order_identification

The work_order_identification specifies ...

4.2.56 Work_request

A Work_request ... The data associated with a Work_request are the following:

- affective_parts;
- approved_by;
- description;
- reason;
- recipients;
- request_date;
- status;
- work_request_identification.

4.2.56.1 affective_parts

The affective_parts specifies ...

4.2.56.2 approved_by

The approved_by specifies ...

4.2.56.3 description

The description specifies ...

4.2.56.4 reason

The reason specifies ...

4.2.56.5 recipients

The recipients specifies ...

4.2.56.6 request_date

The request_date specifies ...

4.2.56.7 status

The status specifies ...

4.2.56.8 work_request_identification

The work_request_identification specifies ...

4.2.57 Analysis_model

An Analysis_model ... The data associated with an Analysis_model are the following:

- actions;
- annotation;
- behaviour_type;
- cell_dimension;
- class;
- data;
- families;
- history;
- model_for;
- physical_dimension;
- refstate;

— units.

4.2.57.1 actions

The actions specifies …

4.2.57.2 annotation

The annotation specifies …

4.2.57.3 behaviour_type

The behaviour_type specifies …

4.2.57.4 cell_dimension

The cell_dimension specifies …

4.2.57.5 class

The class specifies …

4.2.57.6 data

The data specifies …

4.2.57.7 families

The families specifies …

4.2.57.8 history

The history specifies …

4.2.57.9 model_for

The model_for specifies …

4.2.57.10 physical_dimension

The physical_dimension specifies …

4.2.57.11 refstate

The refstate specifies …

4.2.57.12 units

The units specifies …

4.2.58 Analysis_step

An Analysis_step … The data associated with an Analysis_step are the following:

- annotation;
- cell_dimension;
- class;
- data;
- equations;
- families;
- history;
- physical_dimension;
- refstate;
- units;
- zones.

4.2.58.1 annotation

The annotation specifies ...

4.2.58.2 cell_dimension

The cell_dimension specifies ...

4.2.58.3 class

The class specifies ...

4.2.58.4 data

The data specifies ...

4.2.58.5 equations

The equations specifies ...

4.2.58.6 families

The families specifies ...

4.2.58.7 history

The history specifies ...

4.2.58.8 physical_dimension

The physical_dimension specifies ...

4.2.58.9 refstate

The refstate specifies . . .

4.2.58.10 units

The units specifies . . .

4.2.58.11 zones

The zones specifies . . .

4.2.59 Bc_data

A Bc_data . . . The data associated with a Bc_data are the following:

- data_global;
- data_local.

4.2.59.1 data_global

The data_global specifies . . .

4.2.59.2 data_local

The data_local specifies . . .

4.2.60 Bc_dataset

A Bc_dataset . . . The data associated with a Bc_dataset are the following:

- gridloc;
- rstate;
- the_type.

4.2.60.1 gridloc

The gridloc specifies . . .

4.2.60.2 rstate

The rstate specifies . . .

4.2.60.3 the_type

The the_type specifies . . .

4.2.61 Behaviour_model

A Behaviour_model . . . The data associated with a Behaviour_model are the following:

- data;
- dclass;
- dimunits.

4.2.61.1 data

The data specifies . . .

4.2.61.2 dclass

The dclass specifies . . .

4.2.61.3 dimunits

The dimunits specifies . . .

4.2.62 Boundary_condition

A Boundary_condition . . . The data associated with a Boundary_condition are the following:

- datasets;
- family;
- gridloc;
- inward_normal_index;
- inward_normal_list;
- rstate;
- the_type.

4.2.62.1 datasets

The datasets specifies . . .

4.2.62.2 family

The family specifies . . .

4.2.62.3 gridloc

The gridloc specifies . . .

4.2.62.4 inward_normal_index

The inward_normal_index specifies ...

4.2.62.5 inward_normal_list

The inward_normal_list specifies ...

4.2.62.6 rstate

The rstate specifies ...

4.2.62.7 the_type

The the_type specifies ...

4.2.63 Condition

A Condition ... The data associated with a Condition are the following:

— annotation.

4.2.63.1 annotation

The annotation specifies ...

4.2.64 Convergence_history

A Convergence_history ... The data associated with a Convergence_history are the following:

— data;

— iterations;

— norm_definitions.

4.2.64.1 data

The data specifies ...

4.2.64.2 iterations

The iterations specifies ...

4.2.64.3 norm_definitions

The norm_definitions specifies ...

4.2.65 Data_array

A Data_array ... The data associated with a Data_array are the following:

- annotation;
- conversion;
- data;
- data_class;
- data_name;
- dimension;
- exponents;
- sizes;
- units.

4.2.65.1 annotation

The annotation specifies ...

4.2.65.2 conversion

The conversion specifies ...

4.2.65.3 data

The data specifies ...

4.2.65.4 data_class

The data_class specifies ...

4.2.65.5 data_name

The data_name specifies ...

4.2.65.6 dimension

The dimension specifies ...

4.2.65.7 exponents

The exponents specifies ...

4.2.65.8 sizes

The sizes specifies ...

4.2.65.9 units

The units specifies ...

4.2.66 Data_conversion

A Data_conversion . . . The data associated with a Data_conversion are the following:

— offset;

— scale.

4.2.66.1 offset

The offset specifies . . .

4.2.66.2 scale

The scale specifies . . .

4.2.67 Diffusion_equation

A Diffusion_equation . . . The data associated with a Diffusion_equation are the following:

— diffusion_model.

4.2.67.1 diffusion_model

The diffusion_model specifies . . .

4.2.68 Diffusion_model

A Diffusion_model . . . The data associated with a Diffusion_model are the following:

— diff;

— terms.

4.2.68.1 diff

The diff specifies . . .

4.2.68.2 terms

The terms specifies . . .

4.2.69 Dimensional_exponents

A Dimensional_exponents . . .

4.2.70 Dimensional_units

A Dimensional_units . . .

4.2.71 Dirichlet_bc_dataset

A Dirichlet_bc_dataset . . . The data associated with a Dirichlet_bc_dataset are the following:

— dirichlet_data.

4.2.71.1 dirichlet_data

The dirichlet_data specifies . . .

4.2.72 Discrete_data

A Discrete_data . . . The data associated with a Discrete_data are the following:

— data;

— gridloc.

4.2.72.1 data

The data specifies . . .

4.2.72.2 gridloc

The gridloc specifies . . .

4.2.73 Discrete_data_with_rind

A Discrete_data_with_rind . . . The data associated with a Discrete_data_with_rind are the following:

— rind_planes.

4.2.73.1 rind_planes

The rind_planes specifies . . .

4.2.74 Element_list_bc

An Element_list_bc . . . The data associated with an Element_list_bc are the following:

— element_list.

4.2.74.1 element_list

The element_list specifies . . .

4.2.75 Element_range_bc

An Element_range_bc . . . The data associated with an Element_range_bc are the following:

— element_range.

4.2.75.1 element_range

The element_range specifies . . .

4.2.76 Elements_bc

An Elements_bc . . . The data associated with an Elements_bc are the following:

— elements.

4.2.76.1 elements

The elements specifies . . .

4.2.77 Equation

An Equation . . . The data associated with an Equation are the following:

— annotation.

4.2.77.1 annotation

The annotation specifies . . .

4.2.78 Equation_set

An Equation_set . . . The data associated with an Equation_set are the following:

— dclass;

— dimension;

— dimunits;

— equations;

— models.

4.2.78.1 dclass

The dclass specifies . . .

4.2.78.2 dimension

The dimension specifies . . .

4.2.78.3 dimunits

The dimunits specifies . . .

4.2.78.4 equations

The equations specifies . . .

4.2.78.5 models

The models specifies . . .

4.2.79 External

An External . . .

4.2.80 Family

A Family . . . The data associated with a Family are the following:

— annotation;

— conditions;

— geometry.

4.2.80.1 annotation

The annotation specifies . . .

4.2.80.2 conditions

The conditions specifies . . .

4.2.80.3 geometry

The geometry specifies . . .

4.2.81 Gas_model

A Gas_model . . . The data associated with a Gas_model are the following:

— model_type.

4.2.81.1 model_type

The model_type specifies . . .

4.2.82 Geometry_reference

A Geometry_reference . . . The data associated with a Geometry_reference are the following:

— descriptions;

— location.

4.2.82.1 descriptions

The descriptions specifies . . .

4.2.82.2 location

The location specifies ...

4.2.83 Governing_equation

A Governing_equation ... The data associated with a Governing_equation are the following:

- equation_type.

4.2.83.1 equation_type

The equation_type specifies ...

4.2.84 Grid_coordinates

A Grid_coordinates ... The data associated with a Grid_coordinates are the following:

- data;
- descriptions;
- rind.

4.2.84.1 data

The data specifies ...

4.2.84.2 descriptions

The descriptions specifies ...

4.2.84.3 rind

The rind specifies ...

4.2.85 Indices_list

An Indices_list ... The data associated with an Indices_list are the following:

- indices;
- nindices.

4.2.85.1 indices

The indices specifies ...

4.2.85.2 nindices

The nindices specifies ...

4.2.86 Indices_range

An Indices_range . . . The data associated with an Indices_range are the following:

— finish;

— nindices;

— start.

4.2.86.1 finish

The finish specifies . . .

4.2.86.2 nindices

The nindices specifies . . .

4.2.86.3 start

The start specifies . . .

4.2.87 Info

An Info . . . The data associated with an Info are the following:

— description;

— id;

— name.

4.2.87.1 description

The description specifies . . .

4.2.87.2 id

The id specifies . . .

4.2.87.3 name

The name specifies . . .

4.2.88 Integral_data

An Integral_data . . . The data associated with an Integral_data are the following:

— data.

4.2.88.1 data

The data specifies . . .

4.2.89 Matched_mesh_connection

A Matched_mesh_connection … The data associated with a Matched_mesh_connection are the following:

— current;

— donor;

— donor_range;

— range;

— transform.

4.2.89.1 current

The current specifies …

4.2.89.2 donor

The donor specifies …

4.2.89.3 donor_range

The donor_range specifies …

4.2.89.4 range

The range specifies …

4.2.89.5 transform

The transform specifies …

4.2.90 Mesh

A Mesh … The data associated with a Mesh are the following:

— annotation;

— index_count.

4.2.90.1 annotation

The annotation specifies …

4.2.90.2 index_count

The index_count specifies …

4.2.91 Mesh_abutting

A Mesh_abutting . . .

4.2.92 Mesh_cell_data

A Mesh_cell_data . . .

4.2.93 Mesh_connectivity

A Mesh_connectivity . . . The data associated with a Mesh_connectivity are the following:

- annotation;
- current.

4.2.93.1 annotation

The annotation specifies . . .

4.2.93.2 current

The current specifies . . .

4.2.94 Mesh_data

A Mesh_data . . . The data associated with a Mesh_data are the following:

- annotation;
- data;
- the_mesh.

4.2.94.1 annotation

The annotation specifies . . .

4.2.94.2 data

The data specifies . . .

4.2.94.3 the_mesh

The the_mesh specifies . . .

4.2.95 Mesh_overset

A Mesh_overset . . .

4.2.96 Mesh_overset_hole

A Mesh_overset_hole . . .

4.2.97 Mesh_vertex_data

A Mesh_vertex_data . . .

4.2.98 Mismatched_donor_mesh

A Mismatched_donor_mesh . . .

4.2.99 Mismatched_mesh_connection

A Mismatched_mesh_connection . . . The data associated with a Mismatched_mesh_connection are the following:

— gridloc;

— points.

4.2.99.1 gridloc

The gridloc specifies . . .

4.2.99.2 points

The points specifies . . .

4.2.100 Mismatched_mesh_region

A Mismatched_mesh_region . . . The data associated with a Mismatched_mesh_region are the following:

— donor.

4.2.100.1 donor

The donor specifies . . .

4.2.101 Multiple_mesh_block

A Multiple_mesh_block . . . The data associated with a Multiple_mesh_block are the following:

— annotation;

— connectivities.

4.2.101.1 annotation

The annotation specifies . . .

4.2.101.2 connectivities

The connectivities specifies . . .

4.2.102 Neumann_bc_dataset

A Neumann_bc_dataset . . . The data associated with a Neumann_bc_dataset are the following:

— neumann_data.

4.2.102.1 neumann_data

The neumann_data specifies . . .

4.2.103 Point_list_bc

A Point_list_bc . . . The data associated with a Point_list_bc are the following:

— face_center_list_length;

— point_list;

— vertex_list_length.

4.2.103.1 face_center_list_length

The face_center_list_length specifies . . .

4.2.103.2 point_list

The point_list specifies . . .

4.2.103.3 vertex_list_length

The vertex_list_length specifies . . .

4.2.104 Point_range_bc

A Point_range_bc . . . The data associated with a Point_range_bc are the following:

— point_range.

4.2.104.1 point_range

The point_range specifies . . .

4.2.105 Reference_state

A Reference_state . . . The data associated with a Reference_state are the following:

— data;

— state_description.

4.2.105.1 data

The data specifies . . .

4.2.105.2 state_description

The state_description specifies ...

4.2.106 Result

A Result ... The data associated with a Result are the following:

— annotation.

4.2.106.1 annotation

The annotation specifies ...

4.2.107 Rind

A Rind ... The data associated with a Rind are the following:

— index_count;

— planes.

4.2.107.1 index_count

The index_count specifies ...

4.2.107.2 planes

The planes specifies ...

4.2.108 Solution

A Solution ... The data associated with a Solution are the following:

— gridloc;

— solution.

4.2.108.1 gridloc

The gridloc specifies ...

4.2.108.2 solution

The solution specifies ...

4.2.109 Solution_with_rind

A Solution_with_rind ... The data associated with a Solution_with_rind are the following:

— rind_planes.

4.2.109.1 rind_planes

The rind_planes specifies . . .

4.2.110 Structured_donor_mesh

A Structured_donor_mesh . . . The data associated with a Structured_donor_mesh are the following:

— donor;

— points;

— vsize.

4.2.110.1 donor

The donor specifies . . .

4.2.110.2 points

The points specifies . . .

4.2.110.3 vsize

The vsize specifies . . .

4.2.111 Structured_mesh

A Structured_mesh . . . The data associated with a Structured_mesh are the following:

— cell_counts;

— rind_planes;

— vertex_counts.

4.2.111.1 cell_counts

The cell_counts specifies . . .

4.2.111.2 rind_planes

The rind_planes specifies . . .

4.2.111.3 vertex_counts

The vertex_counts specifies . . .

4.2.112 Structured_zone

A Structured_zone . . . The data associated with a Structured_zone are the following:

— grid.

4.2.112.1 grid

The grid specifies ...

4.2.113 Thermal_conductivity_model

A Thermal_conductivity_model ... The data associated with a Thermal_conductivity_model are the following:

— model_type.

4.2.113.1 model_type

The model_type specifies ...

4.2.114 Turbulence_closure

A Turbulence_closure ... The data associated with a Turbulence_closure are the following:

— closure_type.

4.2.114.1 closure_type

The closure_type specifies ...

4.2.115 Turbulence_model

A Turbulence_model ... The data associated with a Turbulence_model are the following:

— diffusion_model;

— model_type.

4.2.115.1 diffusion_model

The diffusion_model specifies ...

4.2.115.2 model_type

The model_type specifies ...

4.2.116 Unstructured_donor_mesh

An Unstructured_donor_mesh ... The data associated with an Unstructured_donor_mesh are the following:

— cells;

— donor;

— interpolant;

— vsize.

4.2.116.1 cells

The cells specifies ...

4.2.116.2 donor

The donor specifies ...

4.2.116.3 interpolant

The interpolant specifies ...

4.2.116.4 vsize

The vsize specifies ...

4.2.117 Unstructured_mesh

An Unstructured_mesh ... The data associated with an Unstructured_mesh are the following:

— cell_count;

— cells.

4.2.117.1 cell_count

The cell_count specifies ...

4.2.117.2 cells

The cells specifies ...

4.2.118 Unstructured_zone

An Unstructured_zone ... The data associated with an Unstructured_zone are the following:

— grid.

4.2.118.1 grid

The grid specifies ...

4.2.119 Vertex_defined_cell

A Vertex_defined_cell ...

4.2.120 Viscosity_model

A Viscosity_model . . . The data associated with a Viscosity_model are the following:

— model_type.

4.2.120.1 model_type

The model_type specifies . . .

4.2.121 Zone

A Zone . . . The data associated with a Zone are the following:

— conditions;

— coordinates;

— dclass;

— dimunits;

— equations;

— family;

— field_data;

— global_data;

— grid;

— grid_connectivity;

— history;

— nindices;

— pyhsical_dimension;

— rstate;

— solution.

4.2.121.1 conditions

The conditions specifies . . .

4.2.121.2 coordinates

The coordinates specifies . . .

4.2.121.3 dclass

The dclass specifies ...

4.2.121.4 dimunits

The dimunits specifies ...

4.2.121.5 equations

The equations specifies ...

4.2.121.6 family

The family specifies ...

4.2.121.7 field_data

The field_data specifies ...

4.2.121.8 global_data

The global_data specifies ...

4.2.121.9 grid

The grid specifies ...

4.2.121.10 grid_connectivity

The grid_connectivity specifies ...

4.2.121.11 history

The history specifies ...

4.2.121.12 nindices

The nindices specifies ...

4.2.121.13 pyhsical_dimension

The pyhsical_dimension specifies ...

4.2.121.14 rstate

The rstate specifies ...

4.2.121.15 solution

The solution specifies ...

4.2.122 Zone_bc

A Zone_bc . . . The data associated with a Zone_bc are the following:

— conditions;

— rstate.

4.2.122.1 conditions

The conditions specifies . . .

4.2.122.2 rstate

The rstate specifies . . .

4.3 Application assertions

This subclause specifies the application assertions for the fluid dynamics data application protocol. Application assertions specify the relationships between application objects, the cardinality of the relationships, and the rules required for the integrity and validity of the application objects and UoFs. The application assertions and their definitions are given below.

TO BE DONE

5 Application interpreted model

5.1 Mapping specification

This clause contains the mapping specification that shows how each UoF and application object of this part of ISO 10303 (see clause 4) maps to one or more AIM constructs (see annex A). Each mapping specifies up to five elements.

> **Application element:** The mapping for each application element is specified in a seperate subclause below. Application object names are given in title case. Attribute names and assertions are listed after the application object to which they belong and are given in lower case.
>
> **AIM element:** The name of one or more AIM entity data types (see annex A), the term "IDENTICAL MAPPING", or the term "PATH". AIM entity data type names are given in lower case. Attributes of AIM entity data types are referred to as <entity name>.<attribute name>. The mapping of an application element may involve more than one AIM element. Each of these AIM elements is presented on a seperate line in the mapping specification. The term "IDENTICAL MAPPING" indicates that both application objects involved in an application assertion map to the same instance of an AIM entity data type. The term "PATH" indicates that the application assertion maps to a collection of related AIM entity instances specified by the entire reference path.
>
> **Source:** For those AIM elements that are interpreted from any common resource, this is the ISO standard number and part number in which the resource is defined. For those AIM elements that are created for the purpose of this part of ISO 10303, this is "ISO 10303–" followed by the number of this part.
>
> **Rules:** One or more global rules may be specified that apply to the population of the AIM entity data types specified as the AIM element or in the reference path. 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. A reference to a global rule may be accompanied by a reference to the subclause in which the rule is defined.
>
> **Reference path:** To describe fully the mapping of an application object, it may be necessary to specify a reference path involving several related AIM elements. Each line in the reference path documents the role of an AIM element relative to the AIM element in the line following it. Two or more such related AIM elements define the interpretation of the integrated resources that satisfies the requirement specified by the application object. For each AIM element that has been created for use within this part of ISO 10303, a reference path 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:

[] enclosed section constrains multiple AIM elements or sections of the reference path are required to satisfy an information requirement;

() enclosed section constrains multiple AIM elements or sections of the reference path are identified as alternatives within the mapping to satisfy an information requirement;

{} enclosed section constrains the reference path to satisfy an information requirement;

<> enclosed section constrains at one or more required reference path;

`||` enclosed section constrains the supertype entity;

`->` attribute references the entity or select type given in the following row;

`<-` entity or select type is referenced by the attribute in the following row;

`[i]` attribute is an aggregation of which a single member is given in the following row;

`[n]` attribute is an aggregation of which member `n` is given 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 string, select, or enumeration type is constrained to a choice or value;

\ the reference path expression continues on the next line;

`*` used in conjunction with braces to indicate that any number of relationship entity data types may be assembled in a relationship tree structure;

`//` enclosed section is an application of one of the mapping templates defined in 5.1.1 below;

`--` the text following is a comment (normally a clause reference).

5.1.1 Mapping templates

This mapping specification includes mapping templates. A mapping template is a reusable portion of a reference path that defines a commonly used part of the structure of the application interpreted model. A mapping template is similar to a programming language macro. The mapping templates used in this part of ISO 10303 are defined in this subclause. Each mapping template definition has three components as follows:

— the template signature that specifies the name of the template and may also specify the names and the order of the formal parameters of the template;

— descriptions of the formal parameters of the template, if any;

— the template body that defines the reusable portion of a reference path and may indicate, through the use of the formal parameter names included in the template signature, the points at which the value parameters are supplied in each template application.

Each mapping template is used at least once in the reference paths specified in **??** to **??**. Each such template application is a reference to the template definition, based on the pattern established by the template signature, and supplies the value parameters that are to be substitued for the formal parameters specified in the template definition. The full reference path can be derived by replacing any formal parameters in the template body by the value parameters specified in the template application and then substituting the completed template body for the template application.

The non-blank characters following the first '/' define the name of the mapping template. The name of the mapping template is given in upper case. The name of the template is followed by a list of parameter values, seperated by commas, enclosed in parentheses. Parameter values are given in lower case except in the case that the value parameter is a string literal that includes upper case characters.

The following notational conventions apply to the definitions and applications of templates:

/ marks the beginning and end of a template signature or a template application;

& prefixes the name of a formal parameter within the definition of a template body;

() enclose the formal parameters in a template signature or the value parameters in a template application;

, separates formal parameters in a template signature or value parameters in a template application;

' ' denotes a string literal that is used as a value parameter in a template application.

Value parameters that are not enclosed by quotes are EXPRESS data type identifiers.

This part of ISO 10303 uses the templates that are specified in the following subclauses.

5.1.1.1 SUBTYPE

The SUBTYPE mapping template specifies a reference to the mapping of a subtype of the current application object. Several such references may be included for one supertype application object.

NOTE This template definition only consists of a template signature, there is no matching template body. The template is included to ease the automatic processing of the mapping specification.

Mapping signature:

/SUBTYPE(application_object)/

Parameter definitions:

application_object: the application object that is a subtype of the current supertype application object and that has the entire or a part of the mapping specification of this supertype.

5.1.1.2 SUPERTYPE

The SUPERTYPE mapping template specifies a reference to the mapping of a supertype of the current application object. Several such references may be included for the subtype application object.

NOTE This template only consists of a signature, there is no matching body. The template is included to ease the automatic processing of the mapping specification.

Mapping signature:

/SUPERTYPE(application_object)/

Parameter definitions:

application_object: the application object that is a supertype of the current subtype application object and that has the entire or a part of the mapping specification of this subtype.

5.2 AIM EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources 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 text for constructs that are imported from the integrated resources. The definitions and EXPRESS provided in the integrated resources for constructs used in the AIM may include select list items and subtypes that are not imported into the AIM. Requirements stated in the integrated resources that refer to select list items and subtypes apply exclusively to those items that are imported into the AIM.

6 Conformance requirements

Conformance to this part of ISO 10303 includes satisfying the requirements stated in this part, 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, ISO 10303-22.

Requirements with respect to implementation methods-specific requirements are specified in annex C.

The Protocol Information Conformance Statement (PICS) proforma lists the options or the combination of options that may be included in the implementation. The PICS proforma is provided in annex D.

This part of ISO 10303 provides for a number of options that may be supported by an implementation. These options have been grouped into the following conformance classes:

— TO BE DONE

Support for a particular conformance class requires support of all the options specified in this class.

Annex A
(normative)
AIM EXPRESS expanded listing

The following EXPRESS is the expanded form of the short form schema given in 5.2. In the event of any discrepancy between the short form and this expanded listing, the expanded listing shall be used.

Annex B
(normative)
AIM short names

Table B.1 provides the short names of entities specified in the AIM of this part of ISO 10303. Requirements on the use of the short names are found in the implementation methods included in ISO 10303.

Annex C
(normative)
Implementation method specific requirements

The implementation method defines what types of exchange behaviour are required with respect to this part of ISO 10303. Conformance to this part of ISO 10303 shall be realized in an exchange structure. The file format shall be encoded according to the syntax and EXPRESS language mapping defined in ISO 10303-21 and in the AIM defined in annex A of this part of ISO 10303. The header of the exchange structure shall identify use of this part of ISO 10303 by the schema name '(TBD — SCHEMA NAME)'.

Annex D
(normative)
Protocol Implementation Conformance Statement (PICS) proforma

This clause lists the optional elements of this part of ISO 10303. An implementation may choose to support any combination of these 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.

Annex E
(normative)
Information object registration

E.1 Document identification

To provide for unambiguous identification of an information object in an open system, the object identifier

{ iso standard 10303 part(237) version(1) }

is assigned to this part of ISO 10303. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

E.2 Schema identification

To provide for unambiguous identification of the schema-name in an open information system, the object identifier

{ iso standard 10303 part(237) version(1) schema(3) aim-short-schema(5) }

is assigned to the **aim_short_schema** schema (see 5.2). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

To provide for unambiguous identification of the schema-name in an open information system, the object identifier

{ iso standard 10303 part(237) version(1) schema(3) aim-long-schema(5) }

is assigned to the **aim_long_schema** schema (see A). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex F
(informative)
Application activity model

The application activity model (AAM) is provided as an aid in understanding the scope and information requirements defined in this application protocol. The model is presented as a set of figures that contain the activity diagrams and a set of definitions of the activities and their data. Activities and data flows that are out of scope are marked with an asterisk.

F.1 Introduction

The application activity model (AAM) is provided as an aid to understanding the scope and information requirements defined in this application protocol. At present, this AAM is restricted in scope to data associated with the process and activities of Computational Fluid Dynamics (CFD) analysis. In the future, the AAM will be extended to include aerodynamic data from ground test and flight test sources.

F.2 Application activity model background

This model describes the overall CFD analysis process, from the perspective of an aerospace manufacturer. The fluid dynamics AP will be developed in stages. The reader is referred to clause 1 for a full description of the initial and ultimate scope of this standard. In its initial state, this part of ISO 10303 defines data standards for laminar, transitional, and turbulent flow of a homogeneous ideal gas. Analyses may be performed on any combination of multi-block structured and/or unstructured grids.

The initial purpose of this part of ISO 10303 is to provide a standard for recording and recovering computer data associated with the numerical solution of the equations of fluid dynamics. The format implemented by this standard is (1) general, (2) portable, (3) expandable, and (4) durable. It is expected that this standard will be extended, in future activity, to provide for storage and retrieval of data from non-analytical sources such as wind tunnel or water tank testing, and flight testing or sea trials.

This part of ISO 10303 consists of a collection of conventions for the storage and retrieval of CFD (computational fluid dynamics) data. The system provides for a standard format for recording the data. A key characteristic of CFD data is that is consists of a relatively small number of very large data arrays. The data format is a conceptual entity established by the documentation intended to do the following:

- Facilitate the exchange of CFD data
  - between sites.
  - between applications codes.
  - across computing platforms.
- Stabilize the archiving of CFD data.

The conventions of this part of ISO 10303 provide for recording a complete and flexible problem description. The exact meaning of a subsonic inflow boundary condition, for example, can be described in complete detail if desired. User comments can be appended nearly anywhere, affording the opportunity, for instance, for date stamping or history information to be included. Dimension and sizing information is carefully defined. Any number of flow variables may be recorded, with or without standard names, and it is also possible to add user-defined or site-specific data. These features afford the opportunity for applications to perform extensive error checking if desired.

Because of this generality, this part of ISO 10303 provides for the recording of much more descriptive information than current applications normally use. However, the provisions for this data are layered so that much of it is optional. It should be practical to convert most current applications to conform to this part of ISO 10303 with little or no conceptual change, retaining the option to take advantage of more detailed descriptions as that becomes desirable.

The specifications of this part of ISO 10303 currently cover the bulk of CFD data that one might wish to exchange among sites or applications; for instance, nearly any type of field data can be recorded, and, based on its name, found and understood by any code that needs it. Global data (e.g., freestream Mach Number, Reynolds number, angle of attack) and physical modeling instructions (e.g., thin layer assumptions, turbulence model) may be specified.

Nevertheless, there are items specific to individual applications for which there is currently no specification within this part of ISO 10303. Most commonly, these are operational instructions, such as number of sweeps, solution method, multigrid directives, and so on. Owing to the miscellaneous nature of this data, there has been no attempt to codify it within a global standard. It is therefore expected that many applications will continue to require small user-generated input files, presumably in ASCII format.

This part of ISO 10303 provides for an extensive set of CFD data. Most applications will make use of only a small subset of this data. Further, inasmuch as applications are viewed as editors that are in the process of building the database, most of them are intended for use on incomplete data sets. Therefore, it is not required that all the data elements specified by these conventions be complete in order for a database to be compliant with this part of ISO 10303. The user must ensure that the current state of the database will support whatever application may be launched. Of course, the application should gracefully handle any absence or deficiency of data. The validity, accuracy and completeness of the data are determined entirely by the applications software.

This part of ISO 10303 serves not only to facilitate the mapping of data onto the compliant file structure but also to standardize the meaning of the recorded data. Thus there are two kinds of conventions operative this standard.

Adherence to the File Mapping: conventions guarantees that conforming software will be able to find and read the data.

Adherence to the Data Definitions: guarantees uniformity of meaning among users and between applications.

This part of ISO 10303 generally avoids the storage of redundant data. Sometimes an application may require an alternate (but intellectually equivalent) form of the data; in such cases it is recommended that the alternate form be prepared at the time of use and kept separate from the Fluid Dynamics AP data file. This avoids habitual reliance on the alternate form, which would invalidate the standard. If the alternate form is appended to the file, care must be taken to update the primary (Fluid Dynamics AP) form whenever permanent changes are made.

F.3 Application activity model definitions

The following terms are used in the application activity model. Terms marked with an asterisk are outside the scope of this application protocol. The definitions given in this annex do not supersede the definitions given in the main body of the text.

F.3.1
Acquire and Modify Geometry
The activity of getting the geometric shape of a product and recasting it in a form suitable for an analysis.

F.3.2
Adaptive Modification to FD Model and Process
The activity of revising the computational model and process.

F.3.3
Adjusted Geometry
Modified geometry that is potentially closer to meeting the requirements.

F.3.4
Adjusted Geometry & Modeling Parameters
Adjusted geometry and modeling parameters revised as a result of a prior analysis.

F.3.5
Analysis data
Data resulting from an analysis.

F.3.6
Analysis Objectives
The desired objectives to be met by an analysis.

F.3.7
Approved Data to Customer
Analysis results intended for transmission to a customer. The results have a formal stamp of approval.

F.3.8
Approved Optimized Geometry
Modified geometry that best meets the requirements. The geometry has a formal stamp of approval.

F.3.9
Approved Results
Results from an analysis that have a formal stamp of approval.

F.3.10
Build and Run Analysis Model
The activity of creating a and running a simulation of a product under particular conditions.

F.3.11
Build FD Computational Model
The activity of creating a simulation of a FD process.

F.3.12
Build Product*

The activity of physically making a product.

F.3.13
Build Prototype*

The activity of physically making a prototype of an intended product.

F.3.14
Compute Flowfield

The activity of computing a FD flowfield.

F.3.15
Computing Implementation

One or more particular computer programs.

F.3.16
Concept

The initial ideas about how a product may be realised.

F.3.17
Conduct Analysis of Design

The activity of simulating the performance of a design of a potential product.

F.3.18
Convergence Error

TBD

F.3.19
Definition of Expected Flow Physics

The expected flow process physics to be simulated.

F.3.20
Definition of Expected Physics

The expected physical aspects to be simulated.

F.3.21
Design, Build, Test and Ship Product

The activity encompassing the the lifecycle stages from designing through delivering a product.

F.3.22
Design Product

The activity of designing a product and generating the information required for it to be made.

F.3.23
Develop Product Design*

The activity of designing a product.

F.3.24
Develop Manufacturing Information*

The activity of generating the information required for a designed product to be made.

F.3.25

Engineering data
TBD

F.3.26
Environment
The product's operating environment which should be simulated.

F.3.27
Error estimates
Estimates of errors in an analysis.

F.3.28
Extent & Environment
For a CFD analysis, the overall physical space to be simulated, and the environment.

F.3.29
Extract FD Engineering Data
The activity of generating engineering-related quantities from FD analysis flowfield data.

F.3.30
Extract Results
The activity of generating engineering-related quantities from analysis result data.

F.3.31
Flow Conditions of Interest
The kind of flow conditions to be analysed.

F.3.32
Flowfield data
Detailed numerical data describing a simulated FD flow.

F.3.33
Geometry
Geometry representing the shape of a product. It may be the result of a design (e.g., a blueprint) or from an analysis.

F.3.34
Grid and Boundary Conditions
The mesh and the boundary conditions used for computing a flowfield.

F.3.35
Manufacturing information*
The information necessary to describe how a product should be built.

F.3.36
Materials*
Physical materials required for building a product.

F.3.37
Mathematical Model
The mathematics forming the basis of a simulation.

F.3.38

Mathematical Model of Fluid Dynamics
The mathematics forming the basis of a FD simulation.

F.3.39
Modeling Parameters
TBD

F.3.40
Practices*
Legal, industrial and company specific practices.

F.3.41
Product*
The physical manifestation of a product.

F.3.42
Prototype*
A potential product or physical model of one.

F.3.43
Optimize Geometry*
The activity of modifying geometry to better meet the requirements.

F.3.44
Optimized Geometry

F.3.45
Resolution Equations
TBD

F.3.46
Requirements
The desired product functionality.

F.3.47
Ship Product*
The activity of shipping a product from its place of manufacture.

F.3.48
Staff & Tools*
Personnel and facilities.

F.3.49
Surface Geometry
The geometry representing the surface of a solid.

F.3.50
Test data*
Data resulting from physical tests.

F.3.51
Test Product*

The activity of physically testing a product.

F.3.52
Test Prototype*

The activity of physically testing a prototype.

F.4 Application Process Description

The application activity model diagrams are given in Figures F.1 through F.6. The graphical form of the application activity model is presented in the IDEF0 activity modeling format. Activities and data flows that are out of scope are shown with dashed lines.

Figures F.1 and F.2 show the general very high level activities related to the design, manufacture, and delivery of a product. Analysis is not apparent at this level.

Figure F.3 shows in more detail the general activities related to the design of a product, one of which is analysing potential designs to estimate their performance against the requirements.

Figure F.4 provides more detail on the activities related to the analysis of a design. At this level, particular analysis processes are not specified. The CFD process is particularised at the next lower level.

The first part of the Computational Fluid Dynamics Process is illustrated in Figure F.5 and consists of a number of process components. At a minimum these include:

A1211: Acquire and Modify Geometry;

A1212: Build FD Computational Model;

A1213: Compute Flowfield.

The second, and final, part of the process is illustrated in Figure F.6. This consists of at least:

A1221: Extract FD Engineering Data;

and may further include:

A1222: Adaptive Modification to FD Model and Process, and/or

A1223: Optimize Geometry.

The Fluid Dynamics AP data standard will be utilized throughout this process, as illustrated in Figure F.7.

The initial source of geometry is a Computer-Aided Design (CAD) system, or a prior analysis which could have been a Finite Element or a CFD analysis, or another source. In many cases, it is necessary to edit the geometry that is provided. This editing may be done to remove complexity from the geometry, and thus reduce the cost and cycle time of the subsequent CFD analysis. In some cases, the geometry may be modified to represent intentionally a model which is different from the baseline model that was represented in the original geometry definition. These modifications, if required, represent the first stage of the analysis process: ‘Acquire and Modify Geometry’ (Process A1211). The product of this process stage is a geometry file representing the surfaces over which fluid may flow.

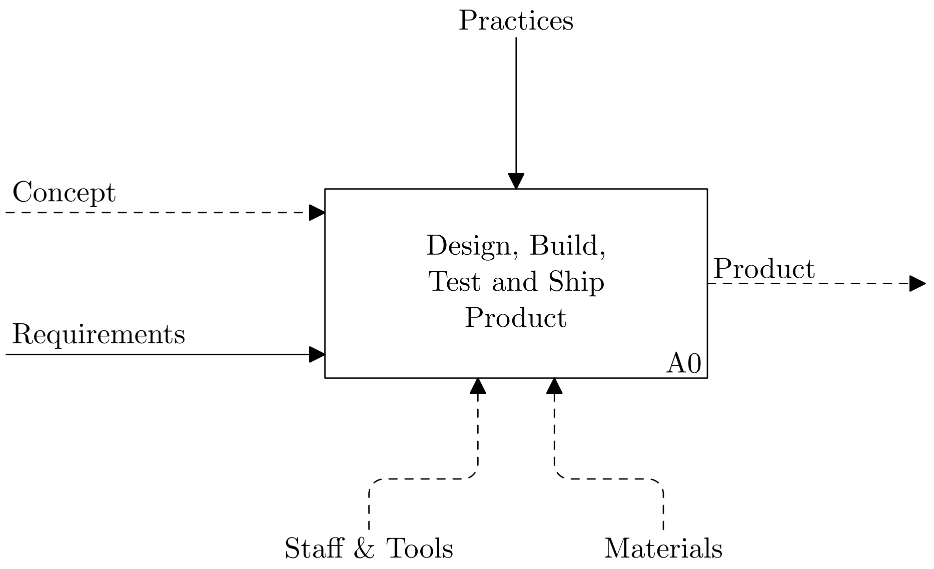

Figure F.1 – A0 Design, build, test and ship product

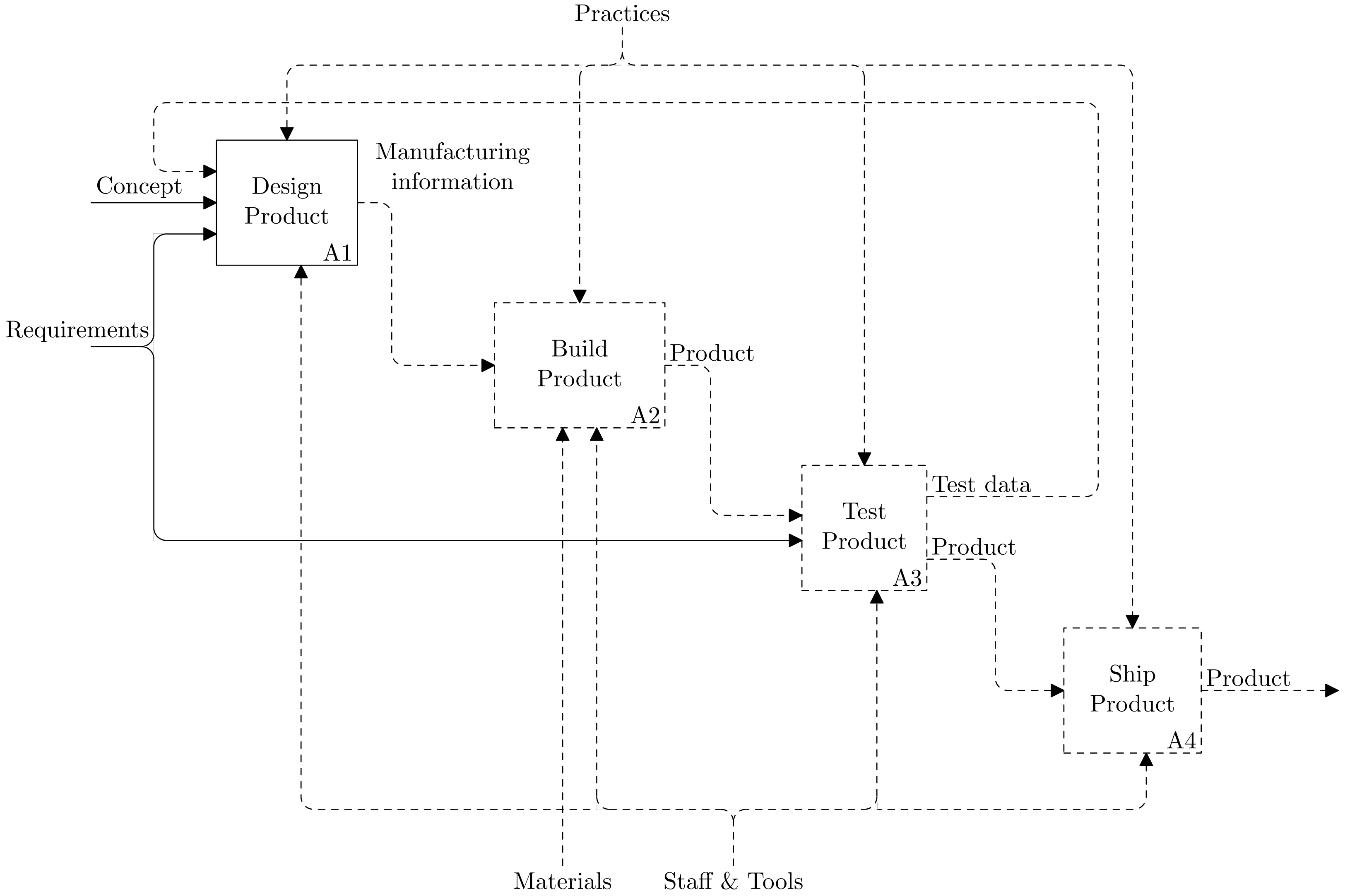

Figure F.2 – A0 Design, build, test and ship product

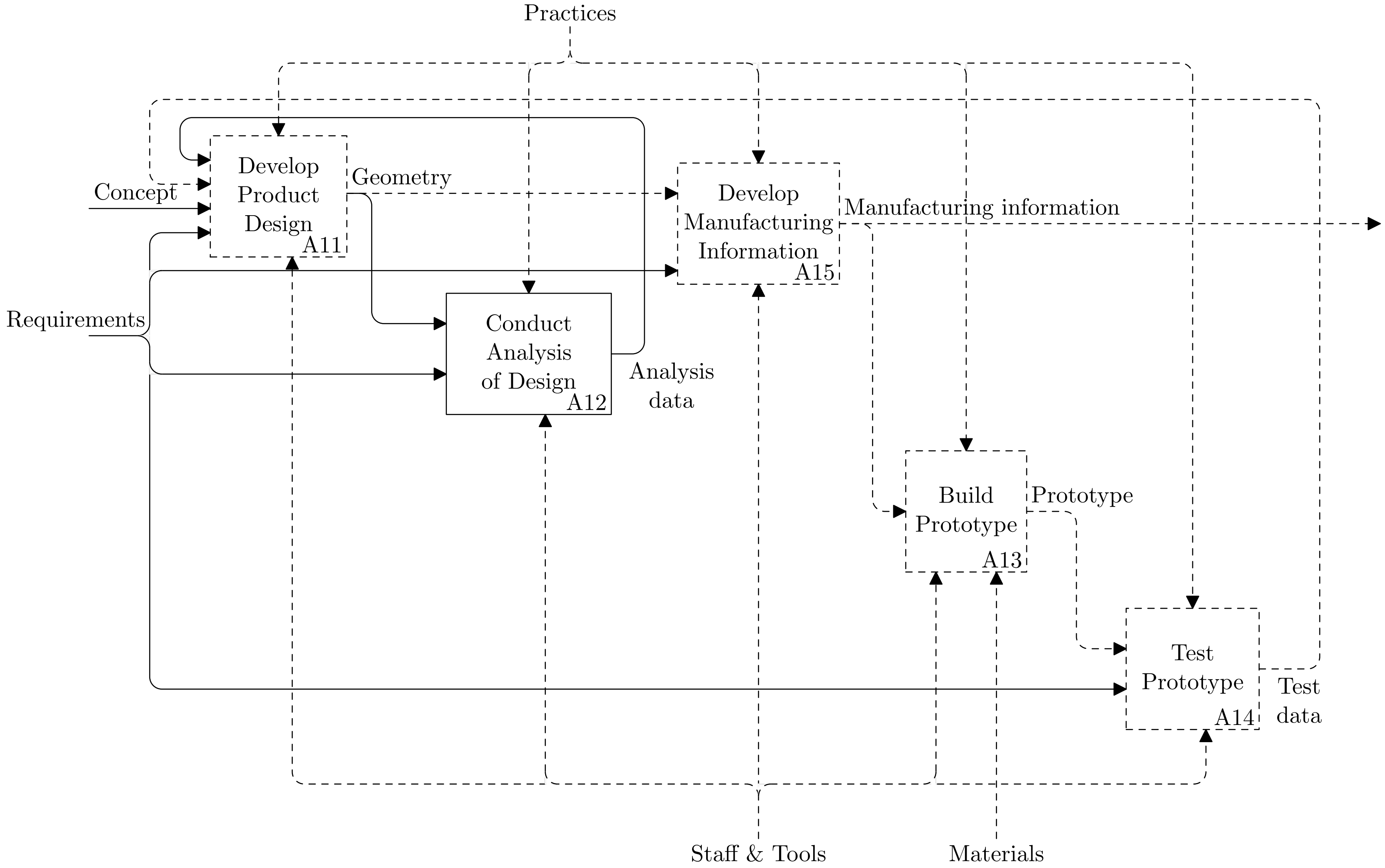

Figure F.3 – A1 Design product

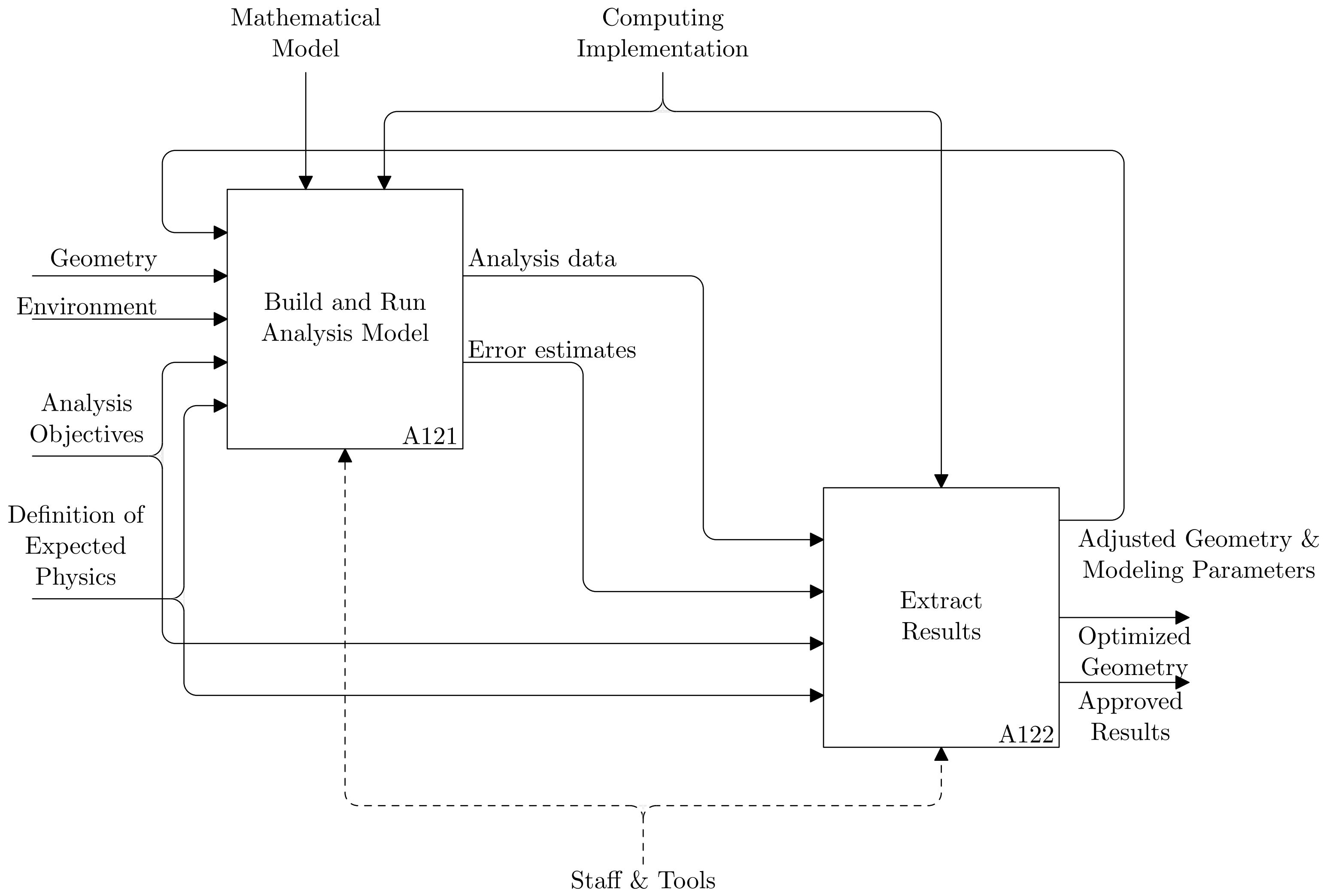

Figure F.4 – A12 Conduct analysis of design

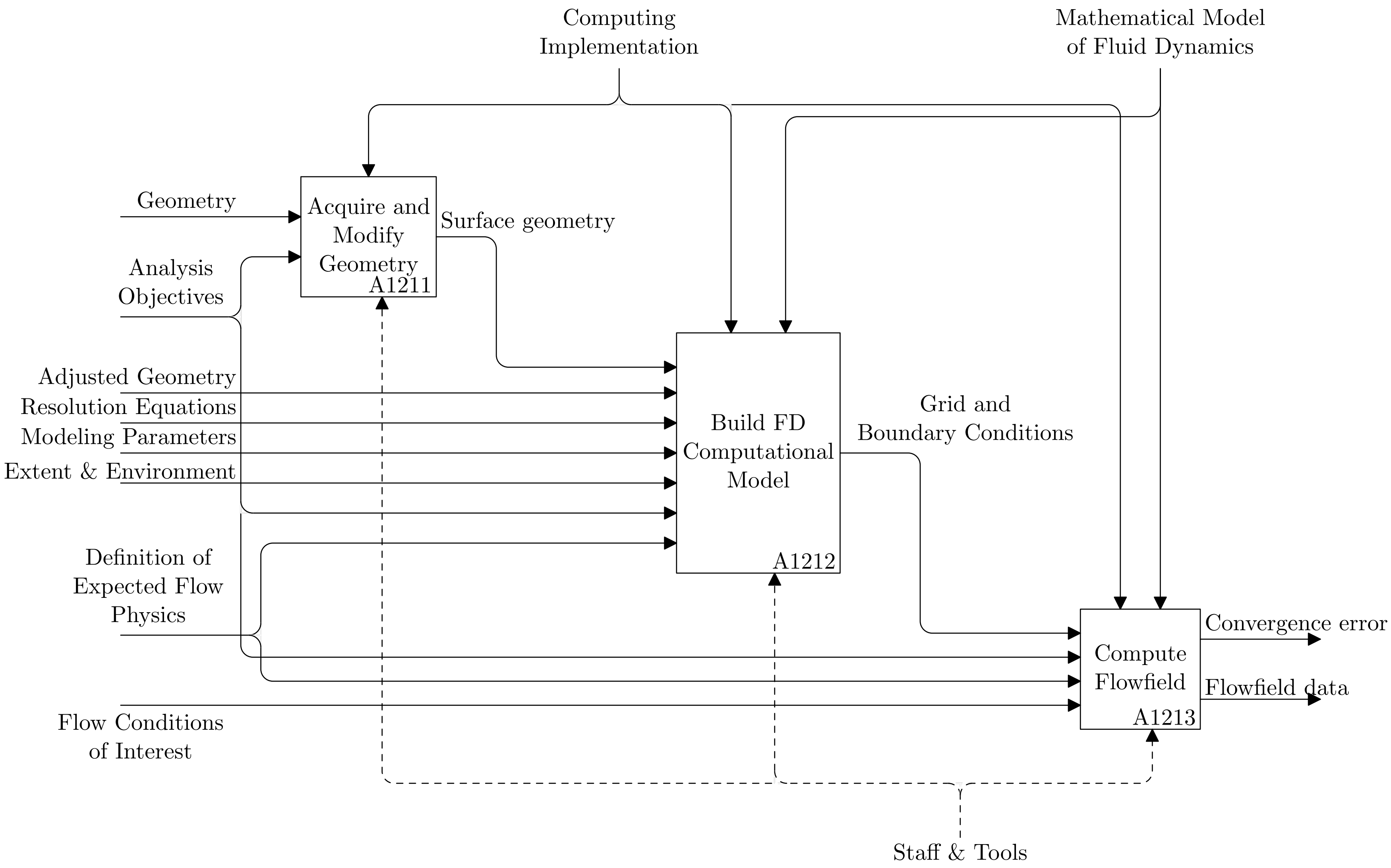

Figure F.5 – A121 Build and run analysis model

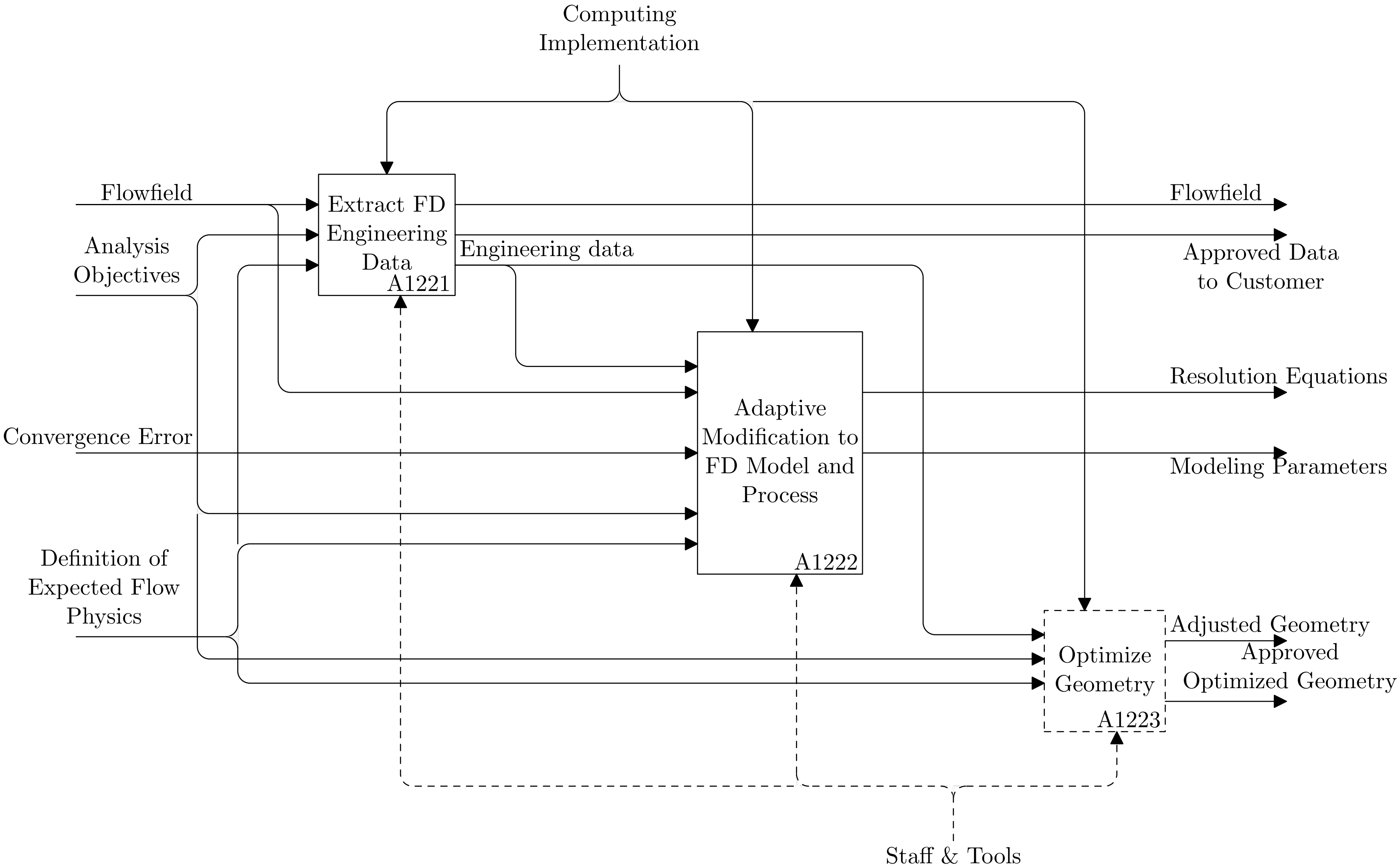

Figure F.6 – A122 Extract results

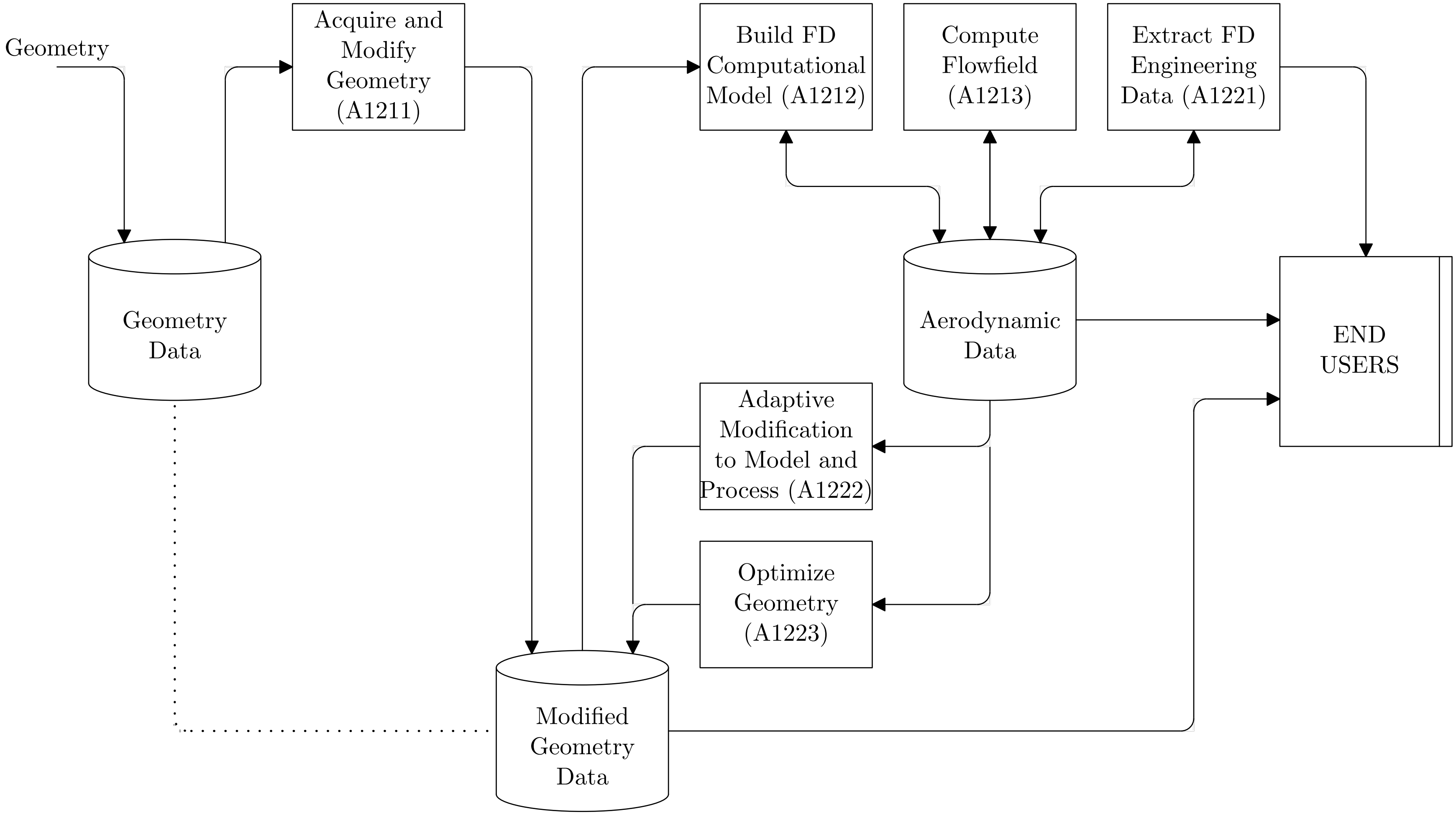
Geometry
Geometry Data
Acquire and Modify Geometry (A1211)
Modified Geometry Data
Build FD Computational Model (A1212)
Compute Flowfield (A1213)
Extract FD Engineering Data (A1221)
Aerodynamic Data
Adaptive Modification to Model and Process (A1222)
Optimize Geometry (A1223)
END USERS

Figure F.7 – Data flow through the CFD process

The CFD analysis process consists of three major stages:

a) Build FD Computational Model (Process A1212) — Using the provided geometry, build the mathematical model and required coordinate systems for the subsequent CFD analysis. The product of this stage is the computational model, represented in the Fluid Dynamics AP format.

b) Compute Flowfield (Process A1213) — Apply the CFD analysis program(s) to the mathematical model to produce the simulation of the aerodynamic flowfield. The flowfield predictions are captured in the Fluid Dynamics AP format.

c) Extract FD Engineering Data (Process A1221) — Extract from the complete flowfield simulation, the data required to meet the end-user's objectives. Typically, these are reduced data such as the net forces which are applied to components of the geometry through the action of the aerodynamic flowfield. The products of this stage of the process may be user-ready data as graphs, printed data, reports, etc., or the products may be additional information which is captured in the Fluid Dynamics AP format.

In some instances, the process also may include one or two recursive loops:

a) Adaptive Modification to FD Model and Process (Process A1222) — In this loop, the characteristics of the flow solver (Process A1213) or the computational model (Process A1212) are modified to improve accuracy or reduce cost of the predictions. These adjustments are controlled by an adaptive algorithm, and by intermediate characteristics of the predicted flowfield. The products of this stage are modifications to the mathematical model and the parameters of the flowfield simulation, stored in the Fluid Dynamics AP format.

b) Optimize Geometry (Process A1223) — In this recursive loop, the analysis geometry is modified based on intermediate flowfield predictions. This geometry modification is made to optimize some attribute of the interaction between the geometry and the flowfield. The products of this stage of the process are: modified geometry; and/or a modified mathematical model or flowfield simulation (Fluid Dynamics AP format).

F.5 Process Variant Approaches

The CFD analysis process includes a number of variant approaches. The relationship of these variant approaches to this part of ISO 10303 is discussed in this subclause.

F.5.1 Equations of Fluid Dynamics

Many different sets of equations of fluid dynamics are used as the basis of a CFD process. Some of the more common mathematical models are known as:

— Navier-Stokes Equations

— Euler Equations

— Nonlinear Potential Flow Equation

— Linear Potential Flow Equation

— Small-Disturbance Equations

- Boundary Layer Equations

- Stream Function Equations

Each of these top-level mathematical models has a number of sub-models, in areas such as computational grid (see F.5.2), flow physics models (see F.5.3), discretization, boundary and initial conditions, and solution algorithm. The CFD analysis process flow is roughly the same for all of these approaches.

F.5.2 Computational Grid

The computational grid provides a local coordinate system for the solution of the mathematical model governing the fluid dynamics problem at issue. Often, the coordinate systems are quite complex, as they as transformed and warped to conform to the details of the geometry to be analyzed. For complex geometries, it is common to introduce multiple independent coordinate systems in subdomains of the flowfield — these subdomains often are called ‘blocks’. Together, all the blocks or subdomains must span the entire flowfield that is to be analyzed.

Several broad types of computational grids are embraced within this part of ISO 10303. These types include:

- Structured grid.

- Structured multi-block 1:1 abutting grid (the coordinate systems are continuous across the boundaries between adjacent blocks).

- Structured multi-block mismatched abutting grid (the coordinate systems are discontinuous across the boundaries between adjacent blocks).

- Structured overset grids (the coordinate systems of adjacent blocks will overlap to a material degree).

- Unstructured tetrahedral grids

- Unstructured prismatic grids

- Unstructured arbitrary N-sided grids.

F.5.3 Flow Physics Models

The equations of fluid dynamics (see F.5.1) describe the conservation of mass, momentum, and energy in a moving fluid. These equations are not adequate, by themselves, to carry out predictions. Additional equations are needed to describe the relationship between thermodynamics properties of the fluid (pressure, temperature, density, viscosity, thermal conductivity, etc). Depending on the requirements of a specific analysis, additional sets of mathematical models may be required to describe the overall impact of turbulence, to describe the impact of chemical reactions such as combustion, and to describe the interaction of an electrically conducting fluid in the presence of electromagnetic fields.

Annex G
(informative)
Application reference model

G.1 Introduction

This annex provides the application reference model for this part of ISO 10303. The application reference model is a representation of the structure and constraints of the application objects specified in clause 4. The application reference model is presented in both EXPRESS and EXPRESS-G. The application reference model is independent from any implementation method. EXPRESS-G is defined in annex D of ISO 10303-11.

The major goal of this ARM is a comprehensive and unambiguous description of the intellectual content of information that must be passed from code to code in a structured-grid multiblock Navier-Stokes analysis system. This information includes grids, flow solutions, multiblock interface connectivity, boundary-conditions, reference states, and dimensional units or normalization associated with data.

The goal is the description of data sets typical of CFD analysis, which tend to contain a small number of extremely large data arrays.

There are two major components to the ARM. The model consists of two schemas and the relationships between these are shown in Figure G.1.

One component, illustrated in Figure G.2, is concerned with the requirements for product data management of CFD analysis data. The elements of this component are based on portions of the ARM in ISO 10303-209.

The second component, illustrated in Figure G.3, is concerned with the requirements of the CFD analysis data itself. This portion of the ARM model was initially based on *CGNS Standard Interface Data Structures* [1], together with extensions proposed in *SIDS additions/modifications to support unstructured meshes and geometry links* [2]. These two documents are now merged into a new version *The CFD General Notation System: Standard Interface Data Structures* [3], and this is the basis for the current model.

G.2 Management arm

The following EXPRESS declaration begins the **management_arm** schema and identifies the necessary external references.

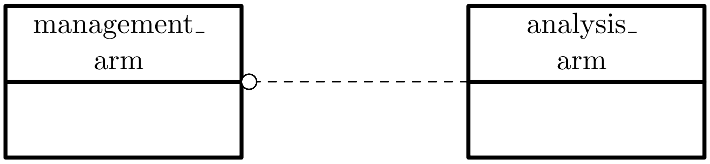

Figure G.1 – Partial schema level ARM diagram (page1 of 1)

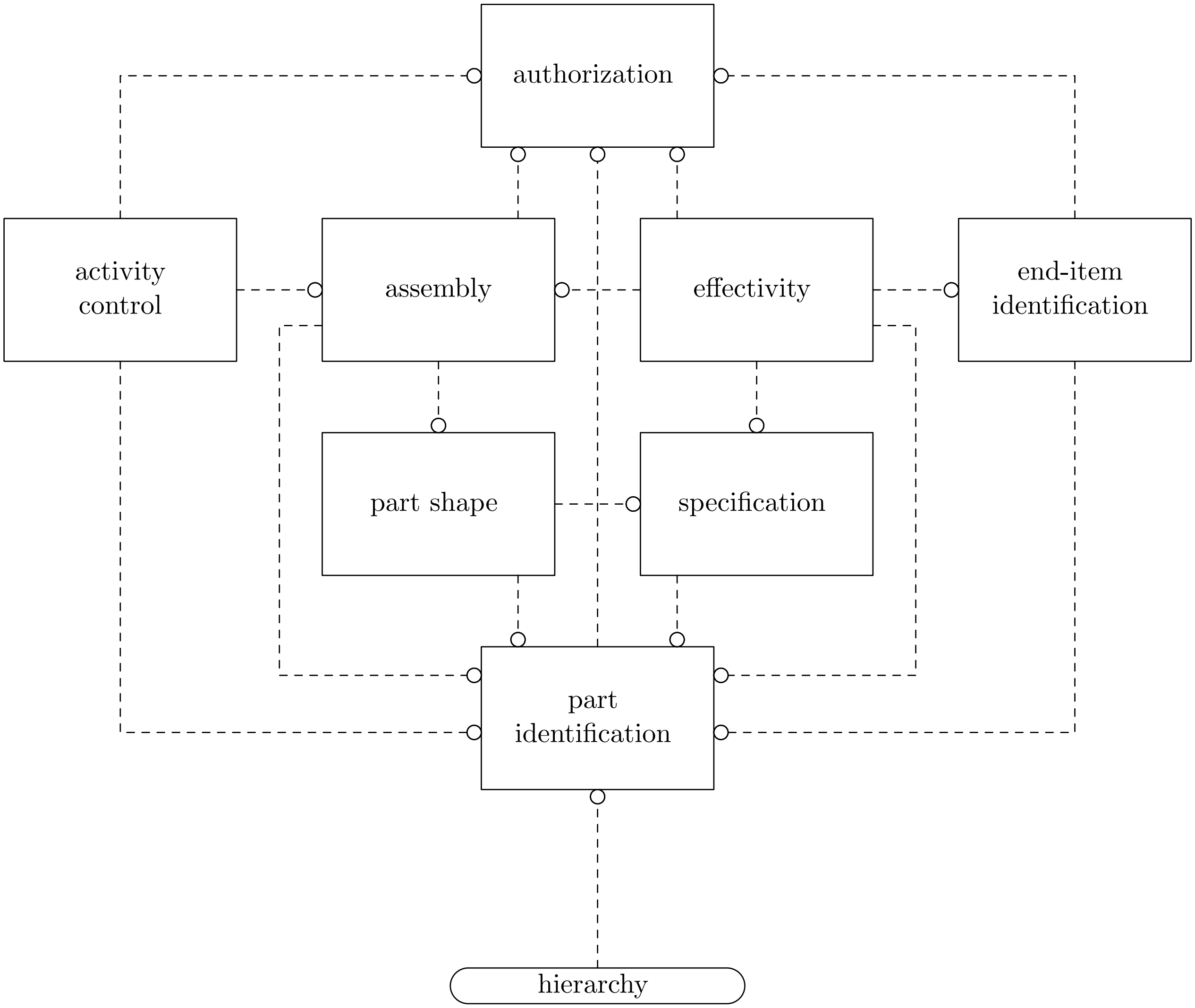

Figure G.2 – Illustration of the major elements for product data management

EXPRESS specification:

```
*)
SCHEMA management_arm;
(*
```

G.2.1 Introduction

This schema defines and describes the structure types for the management of product data related to CFD analyses.

A graphical representation of the schema is shown in Figure G.4 to Figure G.15.

Note to the reader: This schema is almost a straight copy of portions of the ARM for AP 209.

G.2.2 Fundamental concepts and assumptions

The major elements embodied in the schema are illustrated in Figure G.2. In particular the elements in Figure G.2 are represented by the schema constructs as follows:

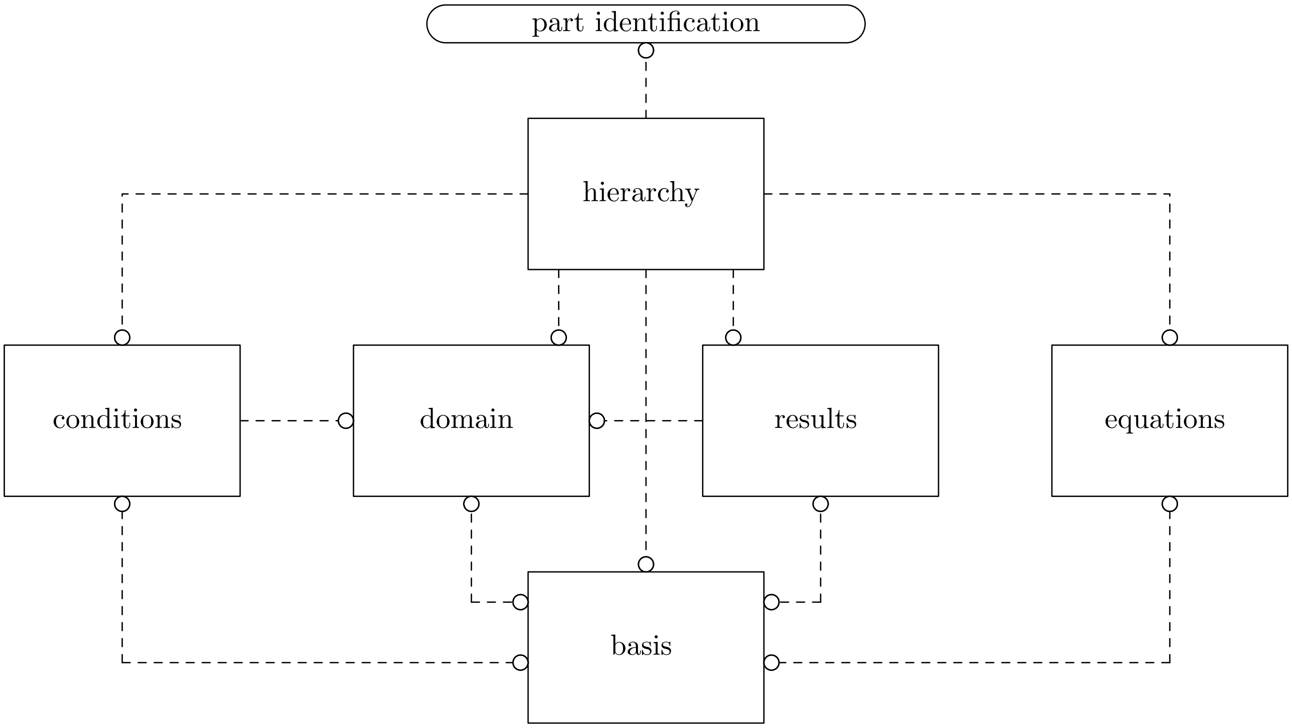

Figure G.3 – Illustration of the major elements for CFD analysis

- Activity control is represented by the structure types shown in Figure G.5;
- Assembly is represented by the structure types shown in Figure G.6 and Figure G.7;
- Authorization is represented by the structure types shown in Figure G.8;
- Effectivity is represented by the structure types shown in Figure G.9 and Figure G.10;
- End-item identification is represented by the structure types shown in Figure G.11;
- Part identification is represented by the structure types shown in Figure G.12;
- Part shape is represented by the structure types shown in Figure G.13 and Figure G.14;
- Specification is represented by the structure types shown in Figure G.15.

G.2.3 management_arm type definitions

G.2.3.1 design_or_analysis

A **design_or_analysis** is

EXPRESS specification:

```
*)
TYPE design_or_analysis = SELECT
  (analysis_version,
```

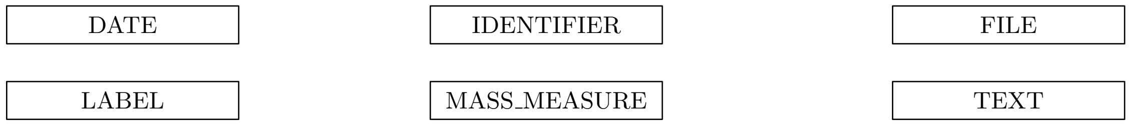

Figure G.4 – Partial entity level diagram of ARM management_arm schema (page 1 of 12)

```
   analysis_discipline_product_definition,
   assembly,
   design_discipline_product_definition,
   part_version);
END_TYPE;
(*
```

G.2.3.2 retention_data_select

A **retention_data_select** is

EXPRESS specification:

```
*)
TYPE retention_data_select = SELECT
  (part,
   design_or_analysis,
   additional_design_information,
   analysis);
END_TYPE;
(*
```

G.2.4 management_arm entity definitions

G.2.4.1 date

A **date** is the identification of a day in a year.

EXPRESS specification:

```
*)
ENTITY DATE;
END_ENTITY;
(*
```

G.2.4.2 identifier

An **identifier** identifies some data.

EXPRESS specification:

```
*)
```

```
ENTITY IDENTIFIER;
END_ENTITY;
(*
```

G.2.4.3 file

A **file** is the identification of a computer file.

EXPRESS specification:

```
*)
ENTITY FILE;
END_ENTITY;
(*
```

G.2.4.4 label

A **label** represents a human-interpretable name of something and has a natural language meaning.

EXPRESS specification:

```
*)
ENTITY LABEL;
END_ENTITY;
(*
```

G.2.4.5 mass_measure

A **mass_measure** is the value of the amount of matter that a body contains.

EXPRESS specification:

```
*)
ENTITY MASS_MEASURE;
END_ENTITY;
(*
```

G.2.4.6 text

A **text** is intended to be read and understood by a human being, for information purposes only.

EXPRESS specification:

```
*)
ENTITY TEXT;
END_ENTITY;
(*
```

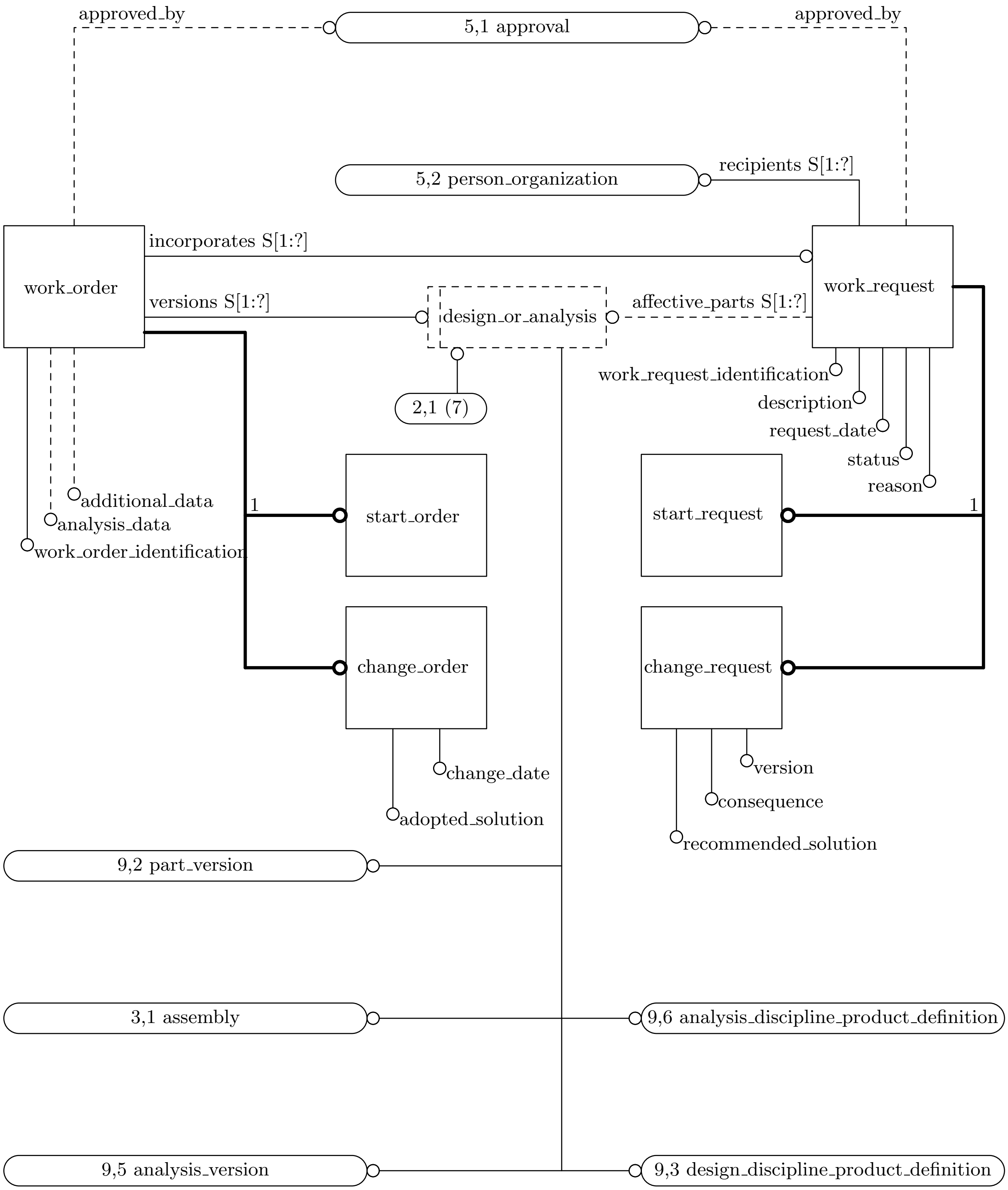

Figure G.5 – Partial entity level diagram of ARM management_arm schema (page 2 of 12)

G.2.4.7 work_order

A **work_order** is a document that authorizes work in the development of an initial or modified design or analysis of a part. It is the result of processing one or more **work_request** objects.

EXPRESS specification:

```
*)
ENTITY work_order;
  additional_data           : OPTIONAL FILE;
  analysis_data             : OPTIONAL FILE;
  approved_by               : OPTIONAL approval;
  incorporates              : SET [1:?] OF work_request;
  versions                  : SET [1:?] OF design_or_analysis;
  work_order_identification : IDENTIFIER;
UNIQUE
  ur1 :work_order_identification;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_work_order FOR work_order;
  ONEOF(start_order,
        change_order);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

work_order_identification: specifies a unique identifier for the **work_order**;

incorporates: specifies the incorporated set of **work_request** objects;

versions: specifies the set of product definition or product versions objects to which the **work_order** applies;

additional_data: specifies additional pertinent information that was compiled during the development of the related set of proposed work requests;

analysis_data: specifies the results of an assessment that was performed to validate the design or modification in the related set of proposed work requests;

approved_by: specifies the approvalof the **work_order**.

G.2.4.8 change_order

A **change_order** is a type of **work_order** that authorizes the development of a modified design or analysis of a part. A **change_order** results in the establishment of a new product definition, **part_version**, or **analysis_version**.

EXPRESS specification:

```
*)
ENTITY change_order
  SUBTYPE OF (work_order);
  change_date      : DATE;
  adopted_solution : TEXT;
END_ENTITY;
(*
```

Attribute definitions:

change_date: specifies the date when the **change_order** became effective;

adopted_solution: specifies the solution selected from the set of recommended solutions.

G.2.4.9 start_order

A **start_order** is a type of **work_order** that authorizes work in the development of the initial design or analysis of a part, resulting in the creation pf an original **part_version** or **analysis_version**.

EXPRESS specification:

```
*)
ENTITY start_order
  SUBTYPE OF (work_order);
END_ENTITY;
(*
```

G.2.4.10 work_request

A **work_request** is a document that identifies the work to be done to initiate or redefine the design or analysis of a part.

EXPRESS specification:

```
*)
ENTITY work_request;
  affective_parts             : OPTIONAL SET [1:?] OF design_or_analysis;
  approved_by                 : OPTIONAL approval;
  description                 : TEXT;
  recipients                  : SET [1:?] OF person_organization;
  reason                      : TEXT;
  request_date                : DATE;
  status                      : LABEL;
  work_request_identification : IDENTIFIER;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_work_request FOR work_request;
  ONEOF(change_request,
        start_request);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

work_request_identification: specifies a unique identifier for the **work_request**;

description: is the explanation of the work to be done;

request_date: specifies the date when the **work_request** was created;

status: specifies the current level of completion of the **work_request**;

reason: specifies the purpose for generating the **work_request**;

affective_parts: specifies the set of product definition or product versions to which the **work_request** applies;

recipients: specifies the set of **person_organization** which receive the **work_request**;

approved_by: specifies the approval of the **work_request**.

G.2.4.11 change_request

A **change_request** is a type of **work_request** that identifies the proposed extent of work to be done to modify the design or analysis of a part.

EXPRESS specification:

```
*)
ENTITY change_request
  SUBTYPE OF (work_request);
  consequence          : TEXT;
  recommended_solution : TEXT;
  version              : LABEL;
END_ENTITY;
(*
```

Attribute definitions:

consequence: specifies the effects in performance, functionality, or form of a particular recommended solution on the affected part or analysis version;

recommended_solution: specifies a potential resolution that would satisfy the requirements deswcribed in the **change_request** through the **description** attribute of **work_request**.

version: specifies a unique identifier for aech iteration of the **change_request**.

G.2.4.12 start_request

A **start_request** is a type of **work_request** that identifies the work to be done to initiate the design or analsis of a **part**.

EXPRESS specification:

```
*)
ENTITY start_request
```

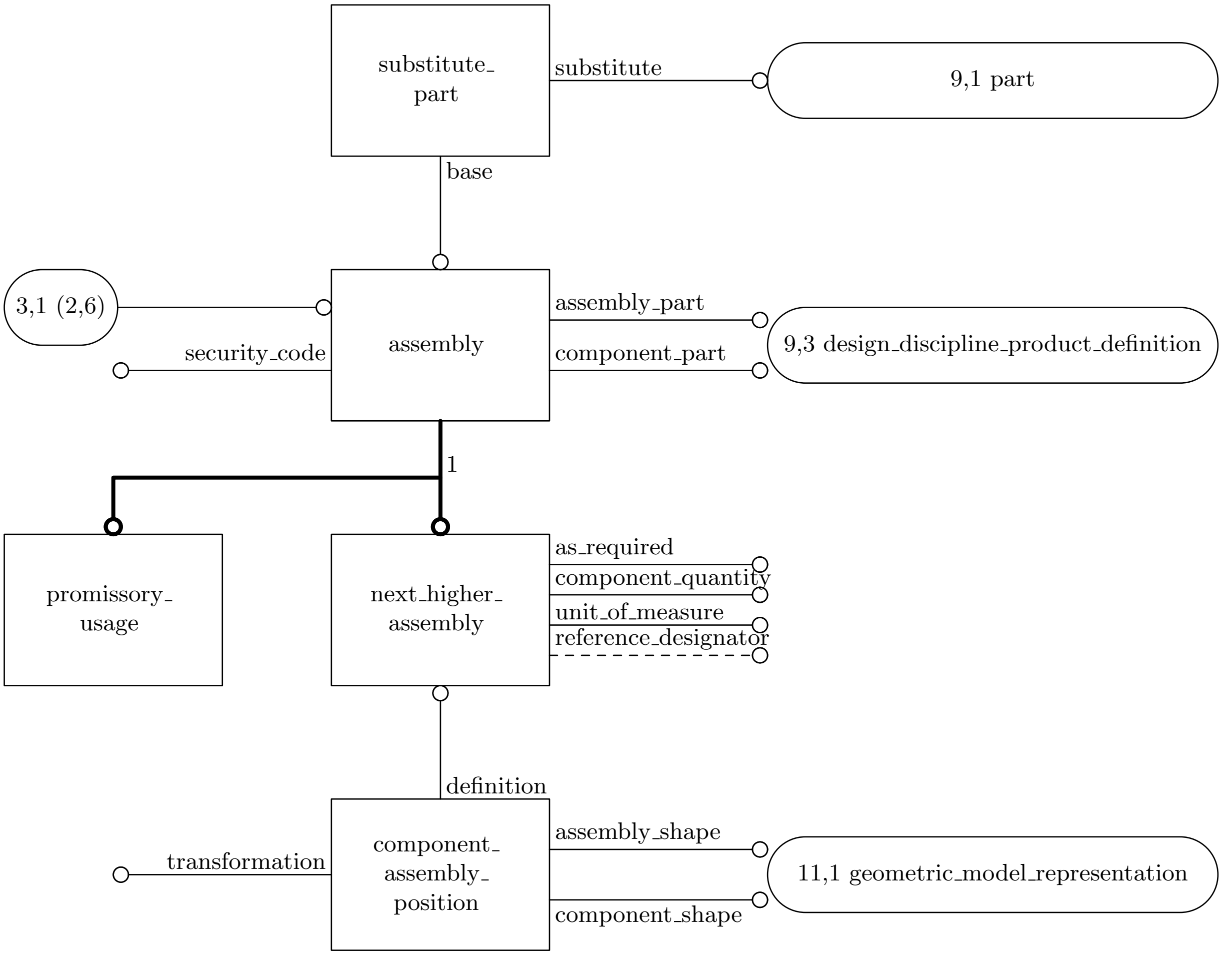

Figure G.6 – Partial entity level diagram of ARM management_arm schema (page 3 of 12)

```
  SUBTYPE OF (work_request);
END_ENTITY;
(*
```

G.2.4.13 assembly

An **assembly** is the parent-child relationship between an assemblage of parts and a component or subassembly used in the assemblage.

EXPRESS specification:

```
*)
ENTITY assembly;
  assembly_part  : design_discipline_product_definition;
  component_part : design_discipline_product_definition;
  security_code  : LABEL;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_assembly FOR assembly;
  ONEOF(promissory_usage,
        next_higher_assembly);
```

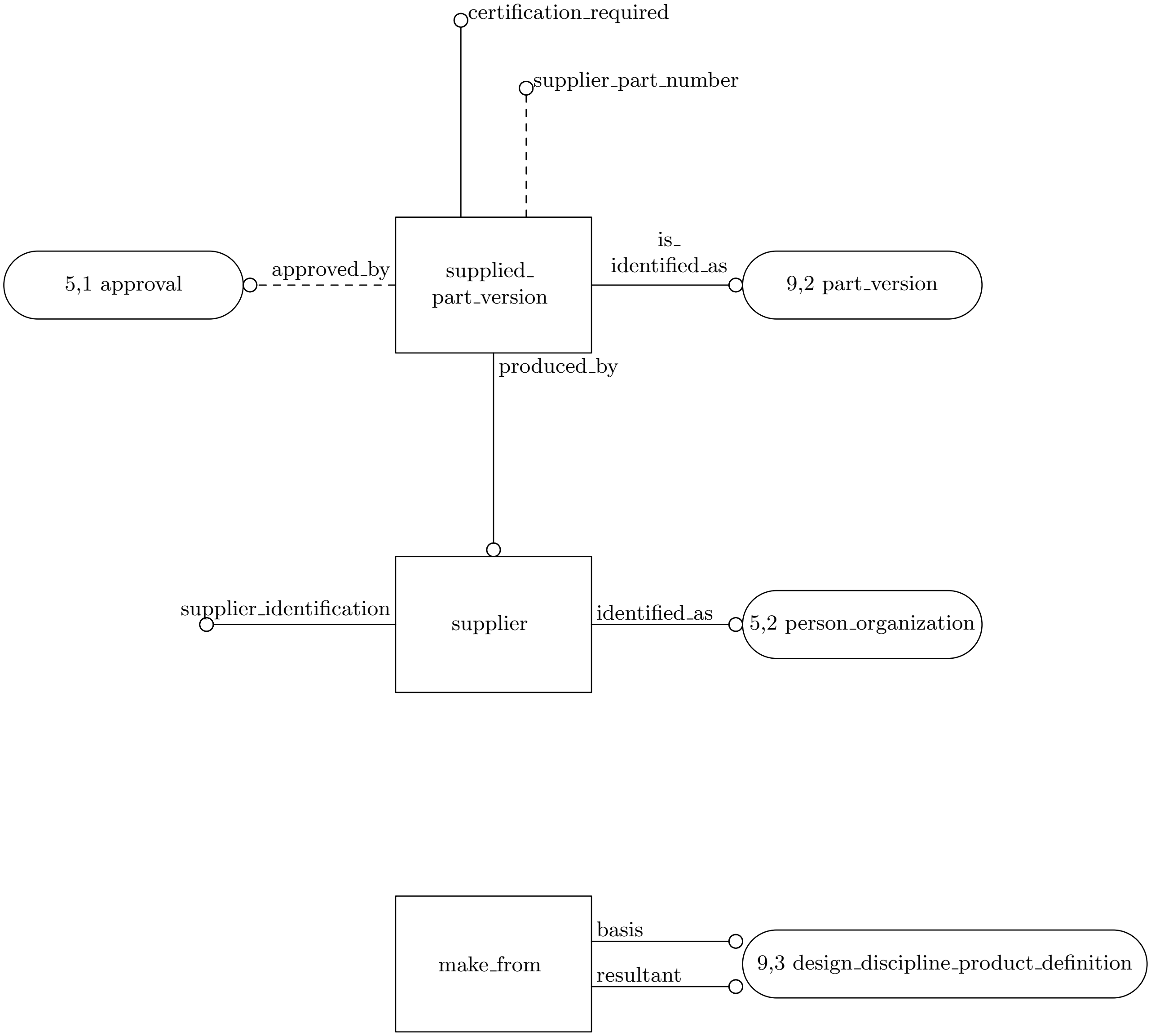

Figure G.7 – Partial entity level diagram of ARM management_arm schema (page 4 of 12)

```
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

security_code: specifies the security classification of the **component_part** within the context of its use in the **assembly**;

assembly_part: specifies the parent product definition in the assmbly relationship. The **assembly_part** may itself be part of a larger assembly.

component_part: specifies the child product definition in the assmbly relationship. The **assembly_part** is an autonomous item that cannot be subdivided further.

G.2.4.14 next_higher_assembly

A **next_higher_assembly** is a type of **assembly** that defines the relationship of a part to its immediate parent within an assembly hierarchy.

EXPRESS specification:

```
*)
ENTITY next_higher_assembly
  SUBTYPE OF (assembly);
  as_required          : LOGICAL;
  component_quantity   : NUMBER;
  reference_designator : OPTIONAL LABEL;
  unit_of_measure      : LABEL;
END_ENTITY;
(*
```

Attribute definitions:

unit_of_measure: if **as_required** is FALSE, specifies the fixed quantity in terms of which the **component_quantity** is expressed;

as_required: specifies whether the component quantity is explicitly defined (FALSE) or is to be quantified as needed in a particualr usage (tree);

component_quantity: if **as_required** is FALSE, specifies the count of a component part in an immediately higher assembly;

reference_designator: if necessary, specifies the unique identifier that distinguishes a particular instance of a component in an assembly where more than one of a particular component part is used in an assembly.

G.2.4.15 promissory_usage

A **promissory_usage** is a type of **assembly** that associates a part to a higher level assembly when the next higher level assembly has not been defined.

EXPRESS specification:

```
*)
ENTITY promissory_usage
  SUBTYPE OF (assembly);
END_ENTITY;
(*
```

G.2.4.16 component_assembly_position

A **component_assembly_position** is the identification of the positioning of a component in a **next_higher_assembly**.

EXPRESS specification:

```
*)
ENTITY component_assembly_position;
  assembly_shape  : geometric_model_representation;
  component_shape : geometric_model_representation;
  definition      : next_higher_assembly;
  transformation  : LABEL;
END_ENTITY;
(*
```

Attribute definitions:

definition: specifies the assembly in which the component is positioned;

assembly_shape: specifies the form of the assembly in which the component is positioned;

component_shape: specifies the form of the component in the assembly;

transformation: specifies translation and orientation of the component in the geometric space of the assembly.

G.2.4.17 substitute_part

A **substitute_part** is a component within an **assembly** whose form, fit, and function might be different from the component it replaces, but that can fulfill the requirements of the latter within the context of the assembly.

EXPRESS specification:

```
*)
ENTITY substitute_part;
  base       : assembly;
  substitute : part;
END_ENTITY;
(*
```

Attribute definitions:

base: specifies the assembly-component relationship in which the substitute may be used;

substitute: specifies the **part** which may be used in place of the base.

G.2.4.18 supplied_part_version

A **supplied_part_version** is a part that is defined by its part number and **Supplier**.

EXPRESS specification:

```
*)
ENTITY supplied_part_version;
```

```
  approved_by            : OPTIONAL approval;
  certification_required : LOGICAL;
  is_identified_as       : part_version;
  produced_by            : supplier;
  supplier_part_number   : OPTIONAL IDENTIFIER;
END_ENTITY;
(*
```

Attribute definitions:

produced_by: specifies the organization that supplies the part;

supplier_part_number: specifies the part number used by the organization that produces the part;

certification_required: specifies whether or not a **supplier** must be approved prior to the procurement of the part from the supplier;

is_identified_as: specifies the **part_version** that the part is used for;

approved_by: specifies the approval of the **supplied_part_version**.

G.2.4.19 supplier

A **supplier** identifies an organization that designs or produces a part.

EXPRESS specification:

```
*)
ENTITY supplier;
  identified_as           : person_organization;
  supplier_identification : IDENTIFIER;
END_ENTITY;
(*
```

Attribute definitions:

supplier_identification: specifies the unique identifier for the supplier;

identified_as: specifies the organization that designs or manufactures a part.

G.2.4.20 make_from

A **make_from** is a relationship between two parts wherein one part is used as the basis for the design of the other part.

EXPRESS specification:

```
*)
ENTITY make_from;
  basis     : design_discipline_product_definition;
  resultant : design_discipline_product_definition;
```

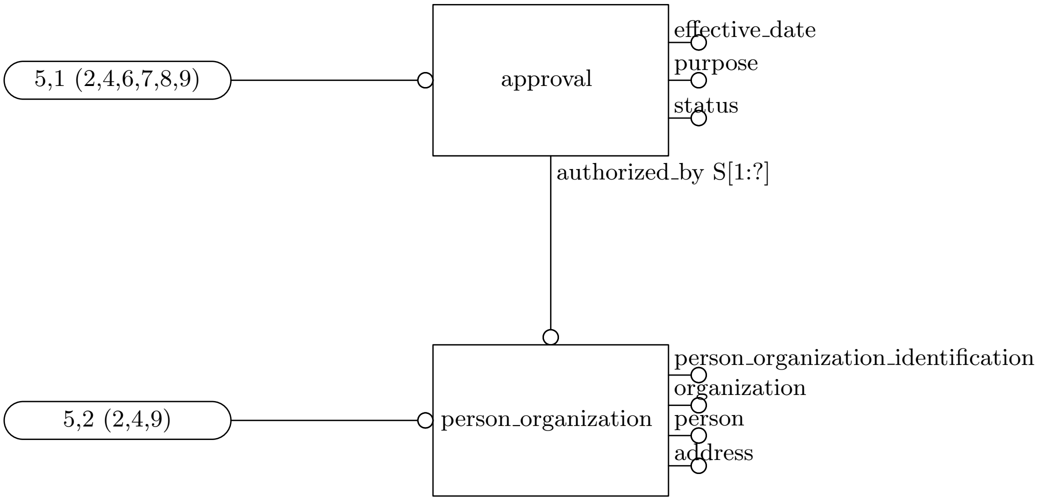

Figure G.8 – Partial entity level diagram of ARM management_arm schema (page 5 of 12)

```
END_ENTITY;
(*
```

Attribute definitions:

basis: specifies the part that serves as the basis for the design of the other part in the relationship;

resultant: specifies the part whose design is based on the other part in the relationship.

G.2.4.21 approval

An **approval** is the indication within an organization of concurrence or nonconcurrence of product data.

EXPRESS specification:

```
*)
ENTITY approval;
  authorized_by  : SET [1:?] OF person_organization;
  effective_date : DATE;
  purpose        : TEXT;
  status         : LABEL;
END_ENTITY;
(*
```

Attribute definitions:

purpose: specifies the reason for the approval;

status: specifies the state of consent applied to product data or a relationship between product data;

effective_date: specifies the date when the approval became, or will become, effective;

authorized_by: the set of **person_organization** authorizing the approval.

G.2.4.22 person_organization

A **person_organization** is the identification of a particular individual and/or organization.

EXPRESS specification:

```
*)
ENTITY person_organization;
  address                            : LABEL;
  organization                       : LABEL;
  person                             : LABEL;
  person_organization_identification : IDENTIFIER;
END_ENTITY;
(*
```

Attribute definitions:

person_organization_identification: specifies a unique identifier for the **person_organization**;

person: specifies an individual;

organization: specifies the collection of people grouped together for one or more common purposes;

address: specifies the routing for paper and electronic mail, or a physical location.

G.2.4.23 effectivity

An **effectivity** is the intended use of a part in a particular configuration of a product.

EXPRESS specification:

```
*)
ENTITY effectivity;
  affected_assemblies : SET [1:?] OF assembly;
  approved_by         : OPTIONAL approval;
  configuration_item  : product_configuration;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_effectivity FOR effectivity;
  ONEOF(sequence_effectivity,
        date_effectivity,
        lot_effectivity);
END_SUBTYPE_CONSTRAINT;
(*
```

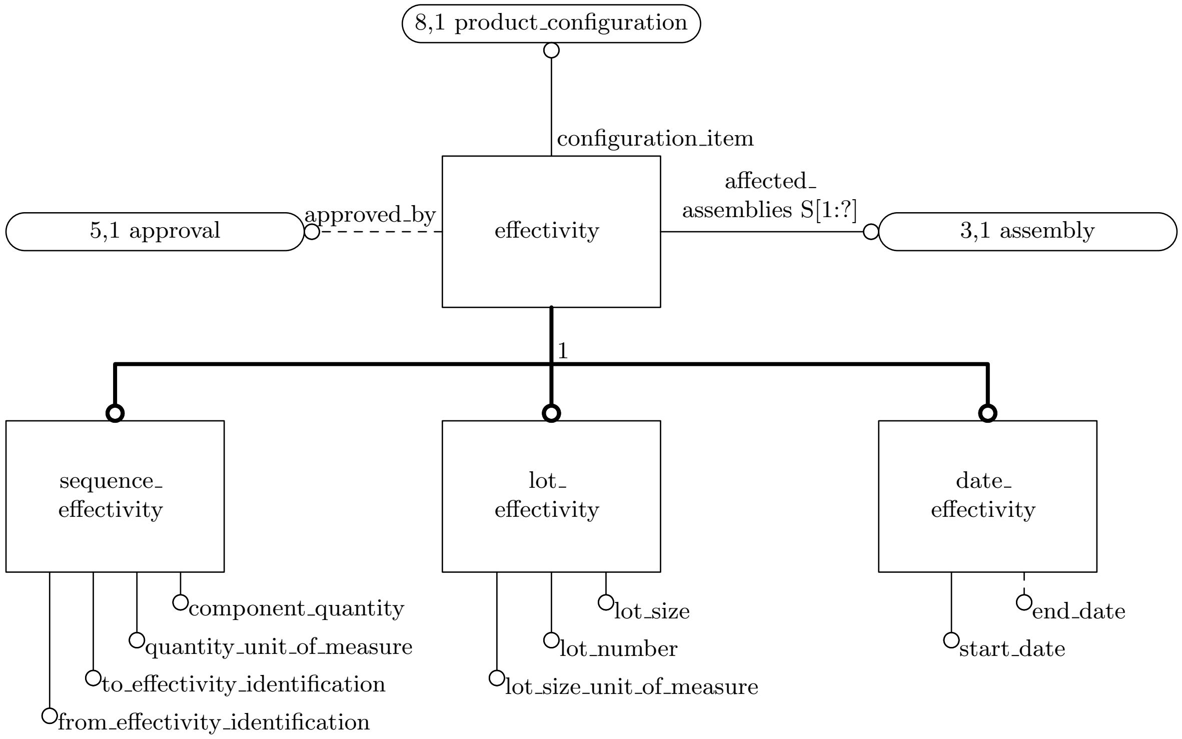

Figure G.9 – Partial entity level diagram of ARM management_arm schema (page 6 of 12)

Attribute definitions:

configuration_item: specifies the **product_configuration** to which the **effectivity** applies;

effected_assemblies: specifies the set of **assembly** objects to which the **effectivity** applies;

approved_by: specifies the approval of the **effectivity**.

G.2.4.24 sequence_effectivity

A **sequence_effectivity** is a type of **effectivity**. It is the specification of the intended use by a design organization of a part within a range of product configurations identified by planned serial numbers.

EXPRESS specification:

```
*)
ENTITY sequence_effectivity
  SUBTYPE OF (effectivity);
  component_quantity              : NUMBER;
  quantity_unit_of_measure        : LABEL;
  from_effectivity_identification : NUMBER;
  to_effectivity_identification   : NUMBER;
END_ENTITY;
(*
```

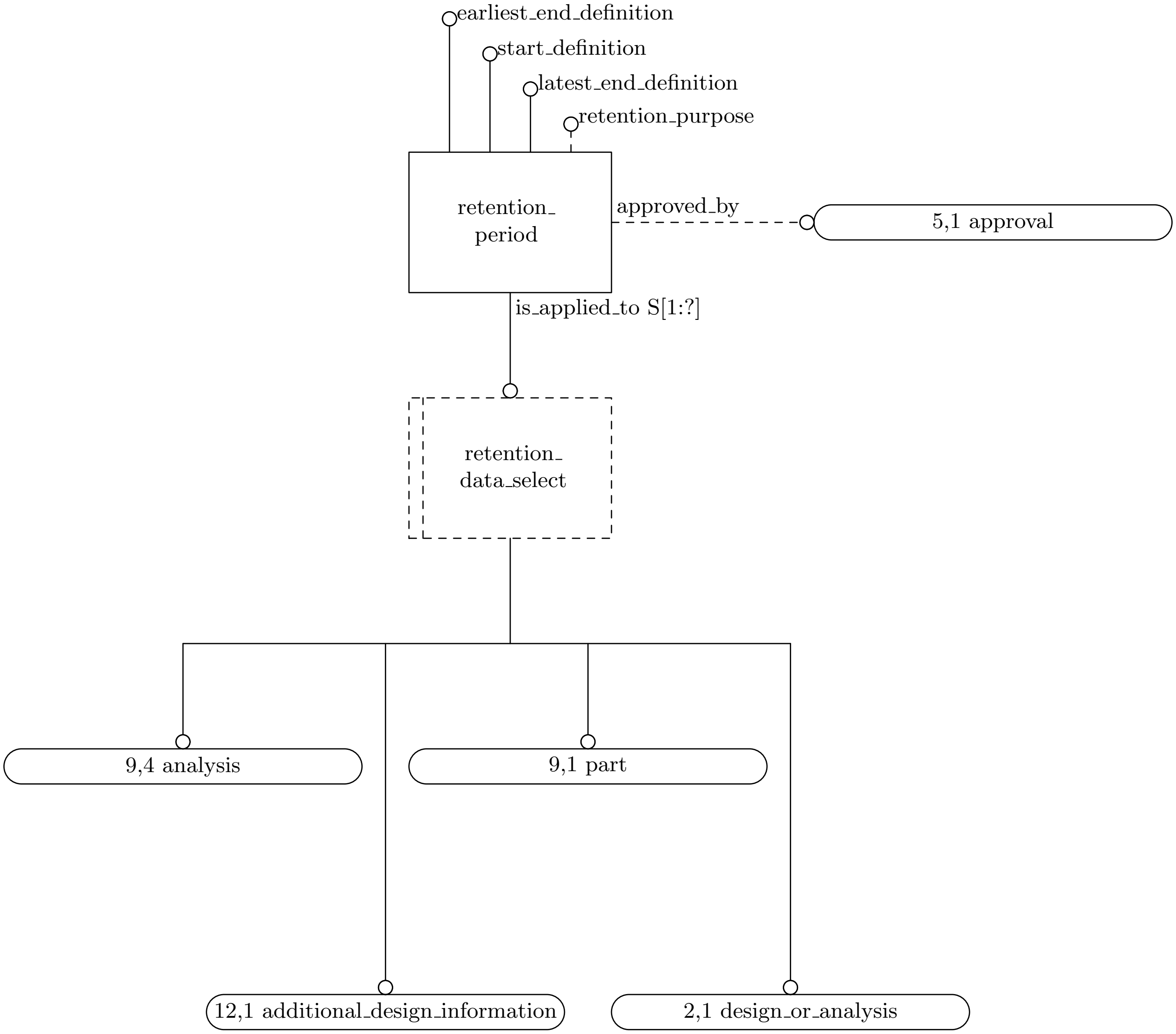

Figure G.10 – Partial entity level diagram of ARM management_arm schema (page 7 of 12)

Attribute definitions:

from_effectivity_identification: specifies the beginning serial number of the range;

to_effectivity_identification: specifies the ending serial number of the range;

component_quantity: specifies the number of units that are effective for a particular part in a configuration;

quantity_unit_of_measure: specifies the measure in which **component_quantity** is expressed.

G.2.4.25 date_effectivity

A **date_effectivity** is a type of **effectivity**. It is the specification by the design organization of the expected usage of a part in a product configuration. The usage of the part within the product configuaration is determined by one or two associated dates.

EXPRESS specification:

```
*)
ENTITY date_effectivity
  SUBTYPE OF (effectivity);
  end_date   : OPTIONAL DATE;
  start_date : DATE;
END_ENTITY;
(*
```

Attribute definitions:

start_date: specifies the first date on which the part is to be used or was used in the **product_configuration**;

end_date: specifies the last date on which the part is planned to be used or was used in the **product_configuration**. An unspecified **date** signifies that an **end_date** has not yet been identified.

G.2.4.26 lot_effectivity

A **lot_effectivity** is a type of **effectivity**. It is a specification of the use of a part in a **product_configuration** where the part is produced as one of a group. Lots are used when parts are produced in batches and/or when important characteristics might vary between production runs.

EXPRESS specification:

```
*)
ENTITY lot_effectivity
  SUBTYPE OF (effectivity);
  lot_number               : NUMBER;
  lot_size                 : NUMBER;
  lot_size_unit_of_measure : LABEL;
END_ENTITY;
(*
```

Attribute definitions:

lot_number: specifies the identification of the group of parts that compose a lot;

lot_number: specifies the

lot_size: specifies the quantity of parts within the lot;

lot_size_unit_of_measure: specifies the fixed quantity amount, in terms of which the **lot_size** is expressed.

G.2.4.27 retention_period

A **retention_period** is the definition of a period of time that product data needs to be maintianed due to organizational policy or legal requirements.

EXPRESS specification:

```
*)
ENTITY retention_period;
  approved_by             : OPTIONAL approval;
  earliest_end_definition : DATE;
  is_applied_to           : SET [1:?] OF retention_data_select;
  latest_end_definition   : DATE;
  retention_purpose       : OPTIONAL TEXT;
  start_definition        : DATE;
END_ENTITY;
(*
```

Attribute definitions:

retention_purpose: specifies the rationale behind the **retention_period**;

is_applied_to: specifies the set of objects whose existence is controlled by the **retention_period**;

start_definition: specifies the time when the **retention_period** starts;

earliest_end_definition: specifies the time at which any object the **retention_period** is applied to may be deleted. In this context, deletion applies to all subordinate objects that are not referernced by other objects.

latest_end_definition: specifies the time at which any object the **retention_period** is applied to shall be deleted. In this context, deletion applies to all subordinate objects that are not referernced by other objects.

approved_by: specifies the approval of the **retention_period**.

G.2.4.28 product_configuration

A **product_configuration** is a variation of a **product_model**. Configuration management is based on this entity.

EXPRESS specification:

```
*)
ENTITY product_configuration;
  approved_by         : OPTIONAL approval;
  item_identification : IDENTIFIER;
  model_name          : product_model;
  parts_configured    : OPTIONAL SET [1:?] OF part;
  phase_of_product    : LABEL;
UNIQUE
  ur1 : item_identification;
END_ENTITY;
(*
```

Attribute definitions:

model_name: specifies the **product_model** that the configuration is a part of;

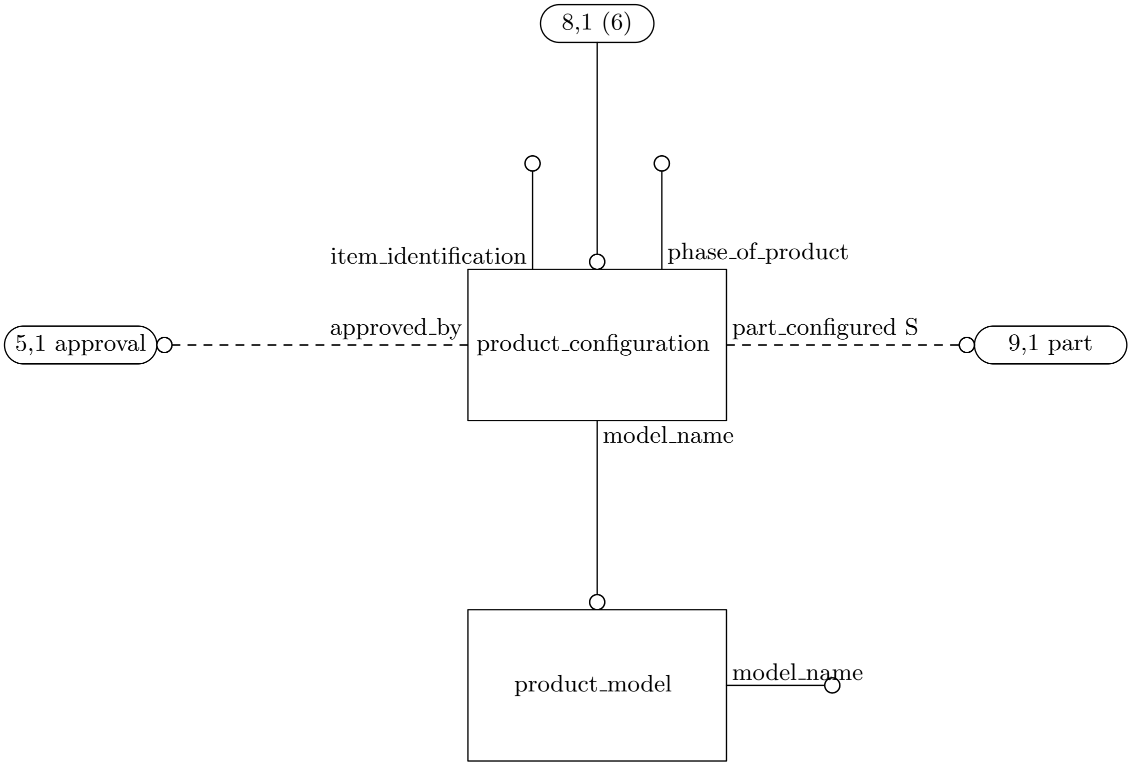

Figure G.11 – Partial entity level diagram of ARM management_arm schema (page 8 of 12)

item_identification: specifies the unique identification of a variation of the **product_model**;

phase_of_product: specifies the stage in the life cycle of the product in which a particular version of the design is planned to be produced;

parts_configured: specifies the set of **part** objects that are being configured;

approved_by: specifies the approval of the **product_configuration**.

G.2.4.29 product_model

A **product_model** is the product the organization provides to its customers. The **product_model** is identified for planning purposes in the design stage of a product.

EXPRESS specification:

```
*)
ENTITY product_model;
  model_name : LABEL;
END_ENTITY;
(*
```

Attribute definitions:

model_name: specifies the unique identification assigned by an organization to a product.

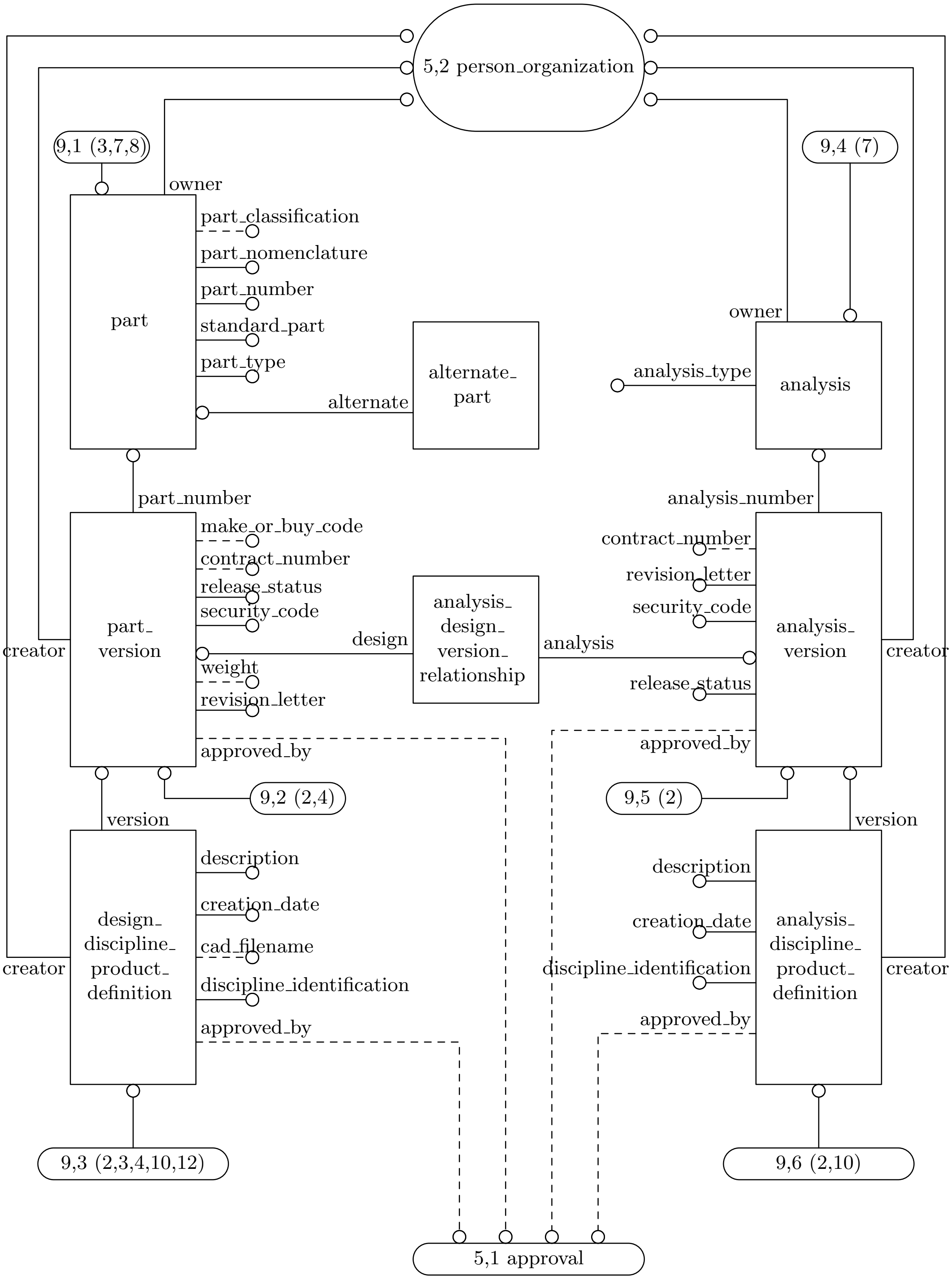

Figure G.12 – Partial entity level diagram of ARM management_arm schema (page 9 of 12)

G.2.4.30 part

A **part** is an item that is intended to be produced or employed in a production process. The item refers not only to the finished part but also to any physical constituent or in-process configuration that makes up the finished part. **part** can also describe stock material where more than one bill-of-material reference information is required.

EXPRESS specification:

```
*)
ENTITY part;
  owner               : person_organization;
  part_classification : OPTIONAL LABEL;
  part_nomenclature   : LABEL;
  part_number         : IDENTIFIER;
  part_type           : LABEL;
  standard_part       : LOGICAL;
UNIQUE
  ur1 : part_number;
END_ENTITY;
(*
```

Attribute definitions:

part_nomenclature: specifes a name by which the **part** is commonly known within an organization. Usually the name reflects the form and/or function of the **part**;

part_number: specifies the unique identification of the **part** for a particular organization;

part_type: specifies one of a set of kinds of parts. It shall always have one out of the four values: 'detail', 'assembly', 'inseperable_assembly' or 'customer_furnished_equipment';

standard_part: specifies whether a part has a design specified externally to the bill-of-material in which an assembly is being defined (TRUE), or it does not have such a design (FALSE);

part_classification: specifies a family of related parts that have common processes applied to them during manufacturing;

owner: the **person_organization** that owns the **part**.

G.2.4.31 alternate_part

An **alternate_part** is a part that is interchangeable with another part with respect to form, fit and function.

EXPRESS specification:

```
*)
ENTITY alternate_part;
  alternate : part;
END_ENTITY;
(*
```

Attribute definitions:

alternate: the **part** that the **alternate_part** can be used for.

G.2.4.32 part_version

A **part_version** is the identification of the representation for a **part** before, during, or after its design has undergone a formal release or change.

EXPRESS specification:

```
*)
ENTITY part_version;
  approved_by      : OPTIONAL approval;
  contract_number  : OPTIONAL IDENTIFIER;
  creator          : person_organization;
  make_or_buy_code : OPTIONAL LABEL;
  part_number      : part;
  release_status   : LABEL;
  revision_letter  : IDENTIFIER;
  security_code    : LABEL;
  weight           : OPTIONAL MASS_MEASURE;
END_ENTITY;
(*
```

Attribute definitions:

part_number: specifies the **part** that the version represents;

revision_letter: specifies the unique identification of the **part_version**;

make_or_buy_code: is optional and if present indicates the design organizations plan for obtaining the part; it may have one of the two values: ‘make’ or ‘buy’;

release_status: specifies the status of the version; it shall always have one of the two values: ‘released’ or ‘unreleased’;

security_code: specifies the security classification of the version;

contract_number: specifies the business contract under which the **part** is designed;

weight: optionally specifies the mass of the version;

creator: specifies the **person_organization** who originated the version;

approved_by: specifies the approval of the **part_version**.

G.2.4.33 analysis_version

An **analysis_version** is the identification of the analysis for a part before, during, or after its analysis has undergone a formal release or change. Only changes that are formally tracked by the responsible organization shall be identified by an **analysis_version**. Those that are not formally tracked shall not be identified by an **analysis_version** but should be tracked by an **analysis_discipline_product_definition**.

EXPRESS specification:

```
*)
ENTITY analysis_version;
  analysis_number : analysis;
  approved_by     : OPTIONAL approval;
  contract_number : OPTIONAL IDENTIFIER;
  creator         : person_organization;
  release_status  : LABEL;
  revision_letter : IDENTIFIER;
  security_code   : LABEL;
END_ENTITY;
(*
```

Attribute definitions:

analysis_number: specifies the unique identification of an **analysis** for a particular organization;

revision_letter: specifies the unique identification of a particular version of an analysis;

release_status: specifies the status of the version of the analysis with respect to the dissemination of analysis information. It shall always have either the value 'released' or the value 'unreleased'. The versions that are released have been reviewed and approved for further use.

security_code: specifies the security classification of a particular version of an analysis;

contract_number: identifies the business contract under which the part is analyzed. An analysis need not be performed under a contract, but if it is then the contract shall be identified.

creator: specifies the **person_organization** that created the **analysis_version**;

approved_by: specifies the approval of the analysis; it need not be specified.

G.2.4.34 analysis_design_version_relationship

An **analysis_design_version_relationship** is a relationship between an **analysis_version** and a **part_version**. The **analysis_version** is with respect to the **part_version**.

EXPRESS specification:

```
*)
ENTITY analysis_design_version_relationship;
  analysis : analysis_version;
  design   : part_version;
END_ENTITY;
(*
```

Attribute definitions:

design: the **part_version**;

analysis: the **analysis_version**.

G.2.4.35 design_discipline_product_definition

A **design_discipline_product_definition** is one of the design organizational definitions or views of a **part_version**. A **design_discipline_product_definition** is controlled by the design organization.

EXPRESS specification:

```
*)
ENTITY design_discipline_product_definition;
  approved_by              : OPTIONAL approval;
  cad_filename             : OPTIONAL FILE;
  creation_date            : DATE;
  creator                  : person_organization;
  description              : TEXT;
  discipline_identification : IDENTIFIER;
  version                  : part_version;
END_ENTITY;
(*
```

Attribute definitions:

version: specifies the **part_version** being defined;

description: specifies the purpose for a particular definition of a product;

cad_filename: specifies the name of the file that contains the geometric description in a computer aided design (CAD) system;

discipline_identification: specifies the identification of the functional unit or group within the organization to which the definition of the product pertains;

creation_date: specifies the date that the **design_discipline_product_definition** was first defined;

creator: specifies the **person_organization** that created the **design_discipline_product_-definition**;

approved_by: specifies approval of the **design_discipline_product_definition**.

G.2.4.36 analysis_discipline_product_definition

An **analysis_discipline_product_definition** is a product definition or view of a **part_version** from the perspective of an analysis organization. The **analysis_discipline_product_definition** is controlled by the analysis organization.

EXPRESS specification:

```
*)
ENTITY analysis_discipline_product_definition;
  approved_by              : OPTIONAL approval;
  creation_date            : DATE;
  creator                  : person_organization;
  description               : TEXT;
  discipline_identification : IDENTIFIER;
  version                   : analysis_version;
END_ENTITY;
(*
```

Attribute definitions:

version: the iteration identified by **analysis_version**;

description: specifies the purpose for a particular definition of the product;

discipline_identification: specifies the identification of the functional unit or group within the organization to which the definition of the product pertains;

creation_date: the date the definition was defined;

creator: the **person_organization** that created the definition;

approved_by: specifies the approval of the definition. It need not be specified.

G.2.4.37 analysis

An **analysis** is an item that is produced as a result of an engineering analysis process.

EXPRESS specification:

```
*)
ENTITY analysis;
  analysis_type : TEXT;
  owner         : person_organization;
END_ENTITY;
(*
```

Attribute definitions:

analysis_type: specifies the type of analysis;

owner: specifies the **person_organization** that owns the **analysis**.

G.2.4.38 geometry_element

A **geometry_element** is a geometry object that defines a portion of a part's shape.

EXPRESS specification:

```
*)
ENTITY geometry_element;
END_ENTITY;
(*
```

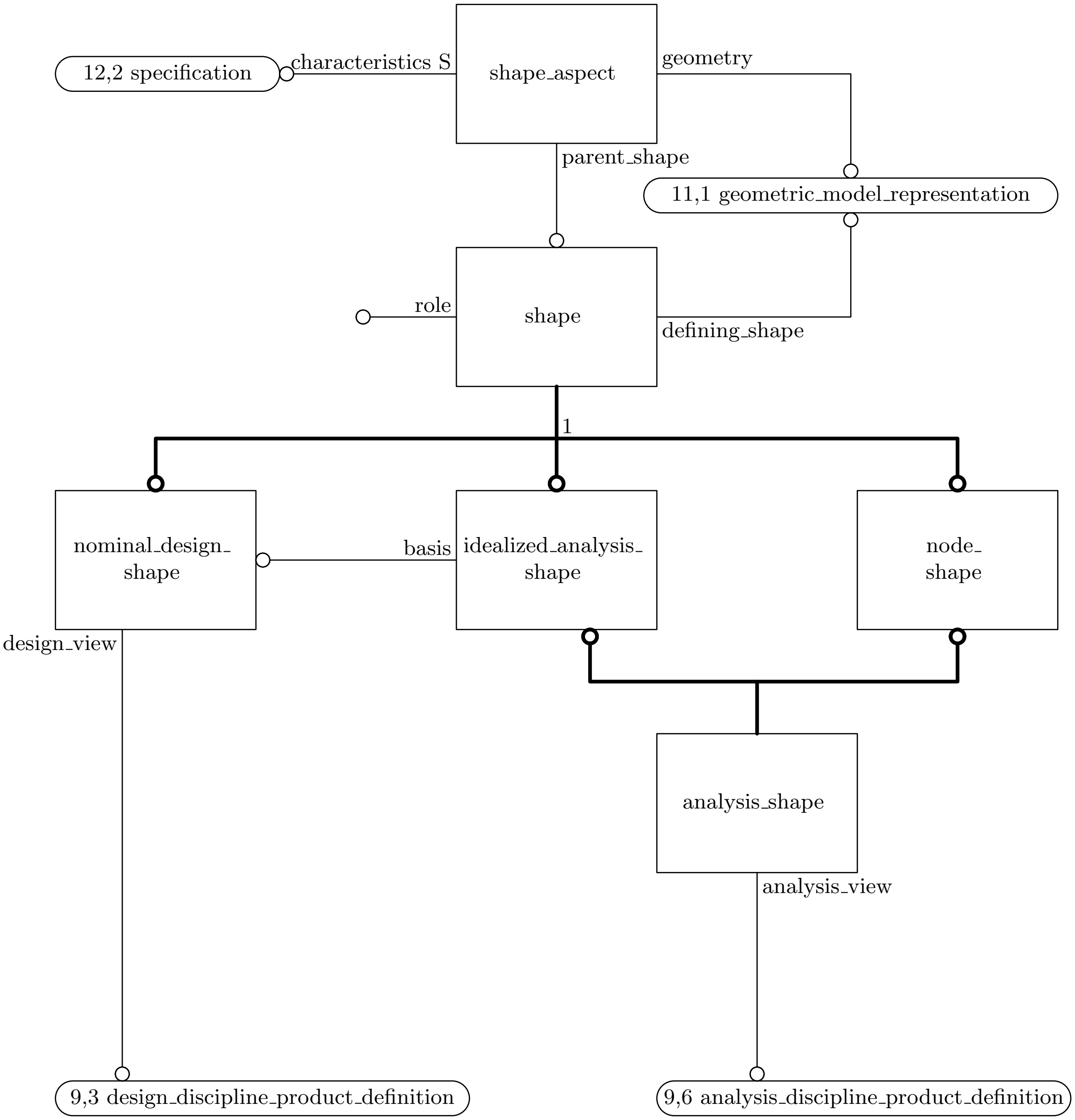

Figure G.13 – Partial entity level diagram of ARM management_arm schema (page 10 of 12)

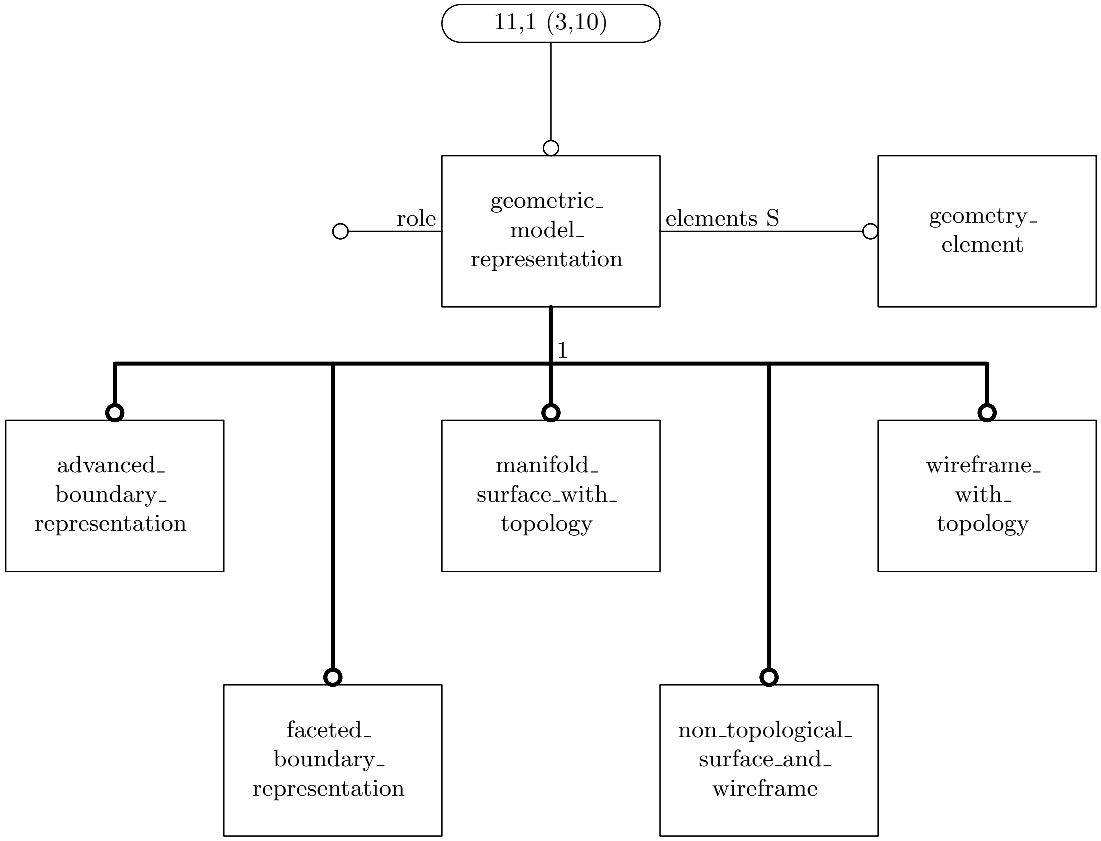
11,1 (3,10)
role
geometric_ model_ representation
elements S
geometry_ element
1
advanced_ boundary_ representation
manifold_ surface_with_ topology
wireframe_ with_ topology
faceted_ boundary_ representation
non_topological_ surface_and_ wireframe

Figure G.14 – Partial entity level diagram of ARM management_arm schema (page 11 of 12)

G.2.4.39 geometric_model_representation

A **geometric_model_representation** defines the shape or a portion of the shape of a part.

EXPRESS specification:

```
*)
ENTITY geometric_model_representation;
  elements : SET [1:?] OF geometry_element;
  role     : TEXT;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_geometric_model_representation FOR
                   geometric_model_representation;
  ONEOF(advanced_boundary_representation,
        faceted_boundary_representation,
        manifold_surface_with_topology,
        non_topological_surface_and_wireframe,
        point_model,
        wireframe_with_topology);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

elements: the component **geometry_element**s;

role: specifies the function that a geometric element plays with respect to the geometric model.

G.2.4.40 advanced_boundary_representation

An **advanced_boundary_representation** is a **geometric_model_representation** that represents a shape by an advanced boundary representation solid model.

EXPRESS specification:

```
*)
ENTITY advanced_boundary_representation
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;
(*
```

G.2.4.41 faceted_boundary_representation

A **faceted_boundary_representation** is a **geometric_model_representation** that represents a shape by a faceted boundary representation solid model.

EXPRESS specification:

```
*)
ENTITY faceted_boundary_representation
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;
(*
```

G.2.4.42 manifold_surface_with_topology

A **manifold_surface_with_topology** is a **geometric_model_representation** that represents a shape using manifold surfaces with topology.

EXPRESS specification:

```
*)
ENTITY manifold_surface_with_topology
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;
(*
```

G.2.4.43 non_topological_surface_and_wireframe

A **non_topological_surface_and_wireframe** is a **geometric_model_representation** that represents a shape using surface or wireframe geometry without topology.

EXPRESS specification:

```
*)
ENTITY non_topological_surface_and_wireframe
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;
(*
```

G.2.4.44 point_model

A **point_model** is a **geometric_model_representation** that is composed of either points or vertices that are defining points of a grid.

EXPRESS specification:

```
*)
ENTITY point_model
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;
(*
```

G.2.4.45 wireframe_with_topology

A **wireframe_with_topology** is a **geometric_model_representation** that represents a prismatic shape using curves representing the shape edge and is topologically constrained.

EXPRESS specification:

```
*)
ENTITY wireframe_with_topology
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;
(*
```

G.2.4.46 shape_aspect

A **shape_aspect** is a distinct portion of a part. The shape of a part consists of one or more **shape_aspect**s.

EXPRESS specification:

```
*)
ENTITY shape_aspect;
  characteristics : SET OF specification;
  geometry        : geometric_model_representation;
  parent_shape    : shape;
END_ENTITY;
(*
```

Attribute definitions:

characteristics: specifies the set of **specification** objects which define qualities or features of the **shape_aspect**;

geometry: specifies the geometric representation for the aspect of the shape;

parent_shape: specifies the **shape** that the **shape_aspect** is a part of.

G.2.4.47 shape

A **shape** is the mathematical representation of the form of a part.

EXPRESS specification:

```
*)
ENTITY shape;
  role : TEXT;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_shape FOR shape;
  ONEOF(nominal_design_shape,
        idealized_analysis_shape,
        node_shape);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

role: the function that the **shape** plays in the definition of a part or analysis model (such as mesh generation geometry);

defining_shape: specifies the geometric model that defines the shape.

G.2.4.48 nominal_design_shape

A **nominal_design_shape** is a type of **shape** that is defined by the design discipline.

EXPRESS specification:

```
*)
ENTITY nominal_design_shape
  SUBTYPE OF (shape);
  defining_shape : geometric_model_representation;
  design_view    : design_discipline_product_definition;
END_ENTITY;
(*
```

Attribute definitions:

defining_shape: is the geometric shape;

design_view: the **design_discipline_product_definition** whose shape is represented by the **nominal_design_shape**.

G.2.4.49 idealized_analysis_shape

An **idealized_analysis_shape** is a type of **shape** and **analysis_shape**. It represents a shape that is suitable a geometric starting shape for an analysis.

EXPRESS specification:

```
*)
ENTITY idealized_analysis_shape
  SUBTYPE OF (shape,
              analysis_shape);
  basis          : nominal_design_shape;
  defining_shape : geometric_model_representation;
END_ENTITY;
(*
```

Attribute definitions:

basis: specifies the **nominal_design_shape** that the model idealizes for analysis purposes;

defining_shape: is the idealized geometric shape;

G.2.4.50 analysis_shape

An **analysis_shape** is a shape for a **part_version** as defined by the analysis discipline.

EXPRESS specification:

```
*)
ENTITY analysis_shape;
  analysis_view : analysis_discipline_product_definition;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_analysis_shape FOR analysis_shape;
  idealized_analysis_shape ANDOR node_shape;
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

analysis_view: the **analysis_discipline_product_definition** whose shape is represented by the **analysis_shape**.

G.2.4.51 node_shape

A **node_shape** is a type of **shape** and **analysis_shape** that is defined by the points of the nodes in a model.

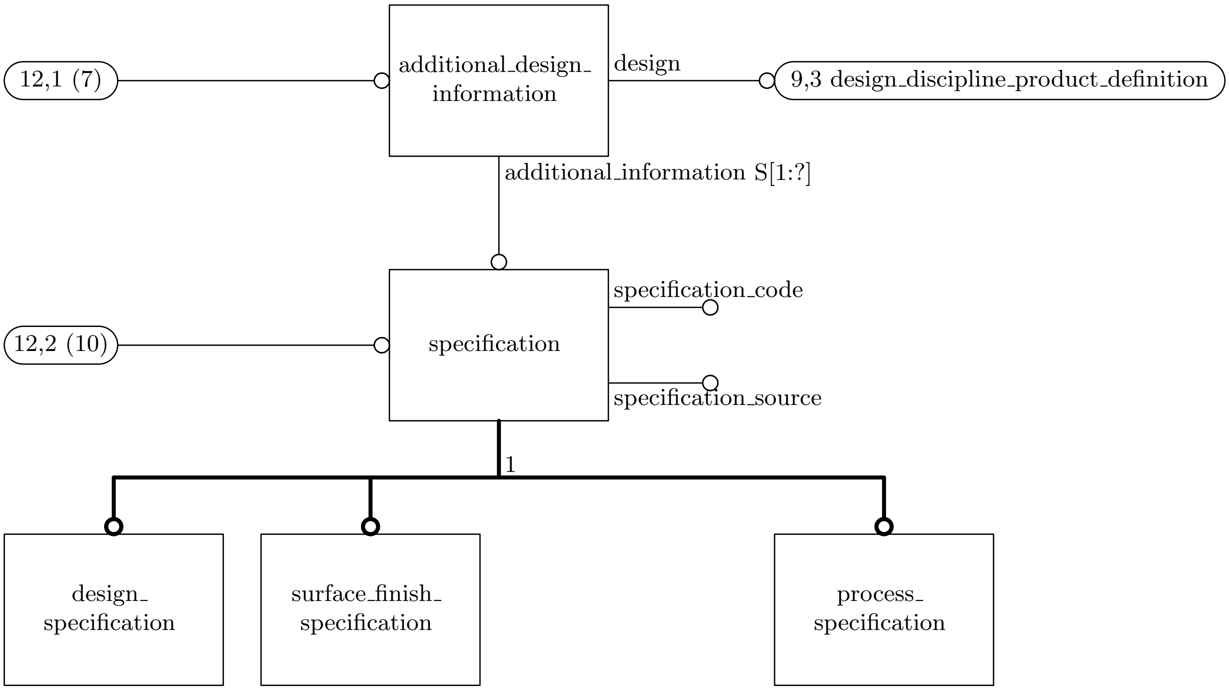
12,1 (7)
additional_design_information
design
9,3 design_discipline_product_definition
additional_information S[1:?]
12,2 (10)
specification
specification_code
specification_source
1
design_specification
surface_finish_specification
process_specification

Figure G.15 – Partial entity level diagram of ARM management_arm schema (page 12 of 12)

EXPRESS specification:

```
*)
ENTITY node_shape
  SUBTYPE OF (shape, analysis_shape);
END_ENTITY;
(*
```

G.2.4.52 additional_design_information

An **additional_design_information** is a collection of specifications that are associated with the design of a part.

EXPRESS specification:

```
*)
ENTITY additional_design_information;
  additional_information : SET [1:?] OF specification;
  design                 : design_discipline_product_definition;
END_ENTITY;
(*
```

Attribute definitions:

design: is the **design_discipline_product_definition** to which the additional information applies;

additional_information: is the set of**specifications** that make up the additional information.

G.2.4.53 specification

A **specification** is a document that contains definitions, processes, or rules related to a unique quality that a finished or in-process part shall posess.

EXPRESS specification:

```
*)
ENTITY specification;
  specification_code   : TEXT;
  specification_source : TEXT;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_specification FOR specification;
  ONEOF(design_specification,
        surface_finish_specification,
        process_specification);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

specification_code: specifies the number or code that identifies the **specification**;

specification_source: specifies the organization that is responsible for the **specification**;

G.2.4.54 design_specification

A **design_specification** is a type of **specification** that sets the design requirements of parts. These requirements are not set by other design or construction features, or referenced data.

EXPRESS specification:

```
*)
ENTITY design_specification
  SUBTYPE OF (specification);
END_ENTITY;
(*
```

G.2.4.55 surface_finish_specification

A **surface_finish_specification** is a type of **specification** that defines properties applicable to surface textures or protective coatings for an in-process or completed part.

EXPRESS specification:

```
*)
```

```
ENTITY surface_finish_specification
  SUBTYPE OF (specification);
END_ENTITY;
(*
```

G.2.4.56 process_specification

A **process_specification** is a type of **specification** that defines a process or service to be performed on a product or material.

EXPRESS specification:

```
*)
ENTITY process_specification
  SUBTYPE OF (specification);
END_ENTITY;
(*
```

EXPRESS specification:

```
*)
END_SCHEMA;  -- end of management_arm schema
(*
```

G.3 Analysis arm

The following EXPRESS declaration begins the **analysis_arm** and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA analysis_arm;
  REFERENCE FROM management_arm
                 (analysis,
                  IDENTIFIER,
                  LABEL,
                  TEXT
                 );
(*
```

NOTE The schemas referenced above can be found in the following parts of ISO 10302:
management_arm this part of ISO 10303

Abbreviated names are used in identifiers of elements declared in this schema. Prefixes used in these identifiers have the following meanings:

cfd computational_fluid_dynamics;

mbna mesh_based_numerical_analysis;

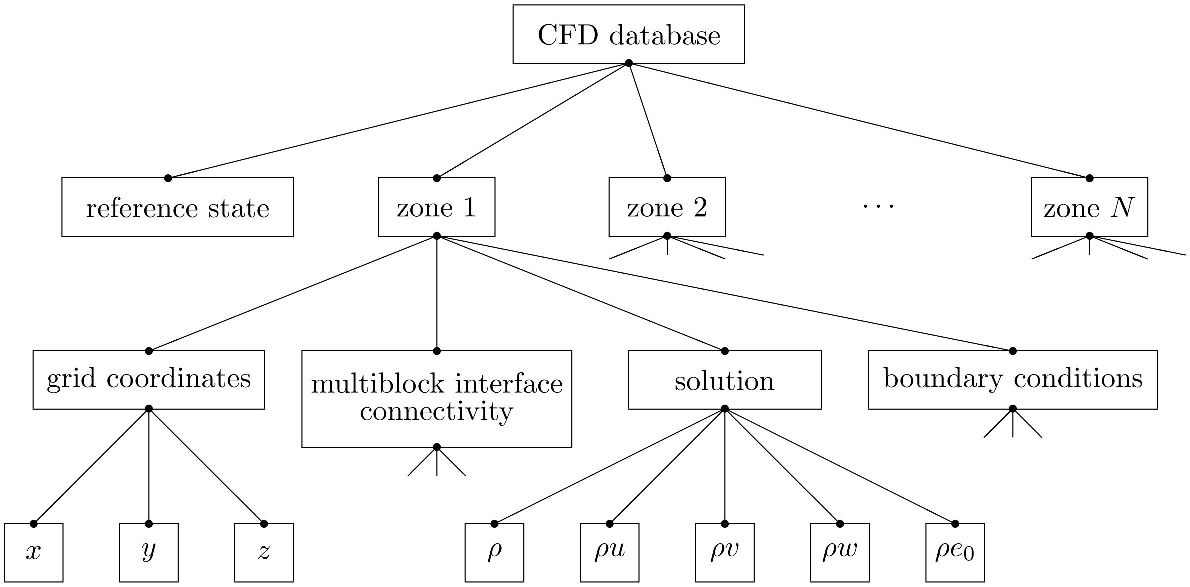

Figure G.16 – Topologically based analysis hierarchy

bc boundary_condition.

Note to the reader: This schema is essentially a concatenation of the schemas in Part 110.

G.3.1 Introduction

This schema defines and describes the top level structures for describing mesh-based CFD numerical analyses.

The major goal of this part of ISO 10303 is a high level, comprehensive and unambiguous description of the intellectual content of information that must be passed from code to code in mesh-based analysis systems. This information includes grids, solutions, multiblock interface connectivity, boundary-conditions, reference states, and dimensional units or normalization associated with data. The data sets typical of these kinds of analysis tend to contain a small number of extremely large data arrays.

The model describes a hierarchical database, precisely defining both the data and their hierarchical relationships.

There are two major alternatives to organizing an analysis hierarchy: topologically based and data-type based. In a topologically based graph, overall organization is by multiblock zones; information pertaining to a particular zone, including its grid coordinates or solution, hangs off the zone. In a data-type based graph, organization is by related data. For example, there would be two nodes at the same level, one for grid coordinates and another for the solution. Hanging off each of these nodes would be separate lists of the zones.

The hierarchy described here is topologically based; a simplified illustration of the database hierarchy is shown in Figure G.16. Hanging off the root 'node' of the database is a node containing global reference-state information, such as general conditions, and a list of nodes for each zone. The figure shows the nodes that hang off the first zone; similar nodes would hang off of each zone in the database. Nodes containing the physical-coordinate data arrays (x, y and z) for the first zone are shown hanging off the 'grid coordinates' node. Likewise, nodes containing the first zone's solution data arrays hang off the 'solution' node. The figure also depicts nodes containing multiblock interface connectivity and boundary condition information for the first zone; subnodes hanging off each of these are not pictured.

Conceptually, an analysis can be thought of as having four components, as in this extremely brief model:

```
ENTITY analysis;
  domain     : computational_space;
  equations  : SET OF equation;
  conditions : SET OF condition;
  results    : SET OF solution;
END_ENTITY;
```

The **domain** is the representation of the computational space (typically a discrete approximation to a continuum). The **equations** are the mathematical formulations of the physics of the problem at hand and the **conditions** are the conditions (e.g., boundary conditions or constraints) that apply to the particular analysis. Finally, the **results** are the calculated solutions to the equations subject to the conditions over the domain, and possibly other data resulting from the analysis.

G.3.2 Fundamental concepts and assumptions

G.3.2.1 Domain

All information pertaining to a given zone, including grid coordinates, solution, and other related data, is contained within that zone's structure entity.

Structures for describing or annotating the database are provided; these same descriptive mechanisms are provided at all levels of the hierarchy.

Certain information, such as units of measure, may be defined at a high level in the hierarchy and apply to all lower levels, unless overriden at one of the lower levels. In turn, the lower level information applies to the subsequent even lower levels, unless overriden again. That is, default information of this kind specified at a low level takes precedence over that specified at a higher level.

A mesh is defined by its vertices and the connections between the vertices. A mesh is a connected graph.

In a structured mesh the cells are arranged in a regular pattern and their shapes are implied by the particular kind of mesh.

A 3–D rectangular mesh is topologically hexahedral. Each cell is a dimensionality 3 topologically hexahedral region defined by eight vertices forming the corners of the hexahedron. Each cell is bounded by six faces, where each face is the quadrilateral defined by four vertices. A face is limited by the four edges that connect the four vertices.

A 2–D rectangular mesh is topologically quadrilateral. Each cell is a dimensionality two topologically quadrilateral region defined by four vertices forming the corners of the quadrilateral. Each cell is limited by the four edges that connect the four vertices.

A 1–D mesh is topologically linear. Each cell is a dimensionality one topologically linear region bounded by two vertices.

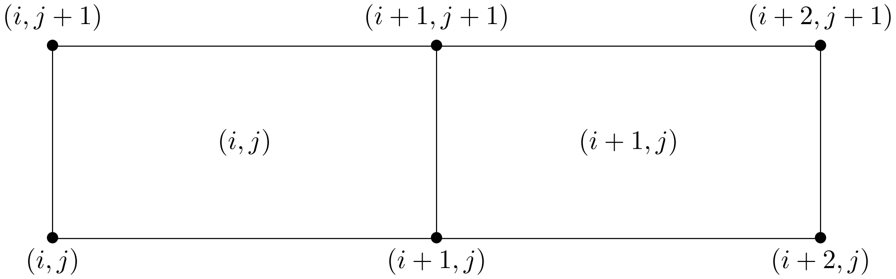

Figure G.17 – Example convention for a 2-D cell center

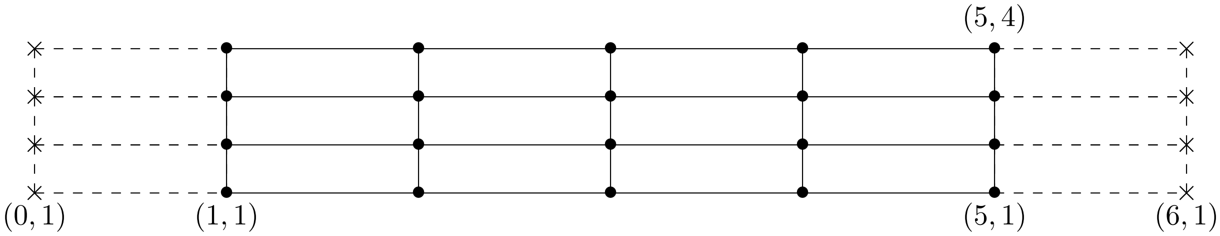

Figure G.18 – Example mesh with rind vertices

Indices describing a structured mesh are ordered: for 3–D (i, j, k); (i, j) is used for 2–D; and (i) for 1–D.

Cell centers, face centers, and edge centers are indexed by the minimum of the connecting vertices.

EXAMPLE 1 For example a 2-D cell center (or face center on a 3-D mesh) would have the conventions shown in Figure G.17.

In addition, the default beginning vertex for a regular mesh is $(1, 1, 1)$; this means the default beginning cell center of a regular mesh is also $(1, 1, 1)$.

There may be locations outside the mesh itself. These are referred to as 'rind' or ghost points and may be associated with fictitious vertices or cell centers. They are distinguished from the vertices and cells making up the mesh (including its boundary vertices), which are referred to as 'core' points.

EXAMPLE 2 Figure G.18 shows a 2–D mesh with a single row of 'rind' vertices at the minimum and maximum i-faces. The mesh size (i.e., the number of 'core' vertices in each direction) is 5×4. 'Core' vertices are designated by '•', and 'rind' vertices by '×'. Default indexing is also shown for the vertices.

For a mesh, the minimum faces in each coordinate direction are denoted i-min, j-min and k-min; the maximum faces are denoted i-max, j-max and k-max. These are the minimum and maximum 'core' faces.

EXAMPLE 3 $i-$min is the face or grid plane whose core vertices have minimum i index (which if using default indexing is 1).

An unstructured mesh is composed of vertices and cells, where the cells need not form a regular pattern and the shape of the cells is not restricted to be uniform throughout the mesh. Cells have vertices at their corners and may also have vertices (nodes) on the edges, faces, and interior of the cell.

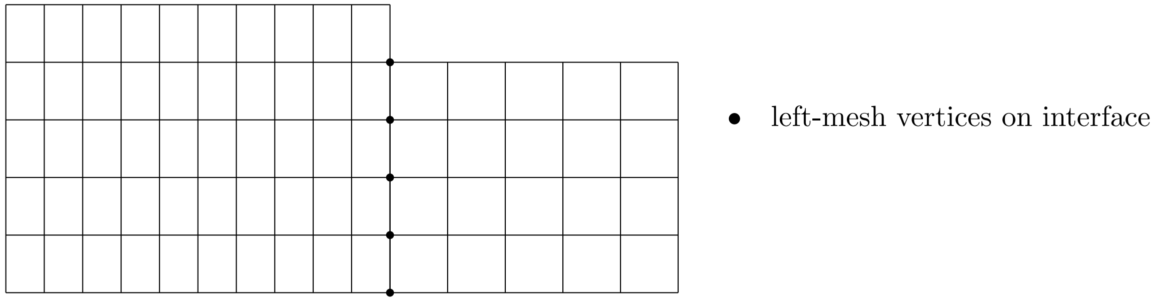

Figure G.19 – A 1-to-1 abutting interface

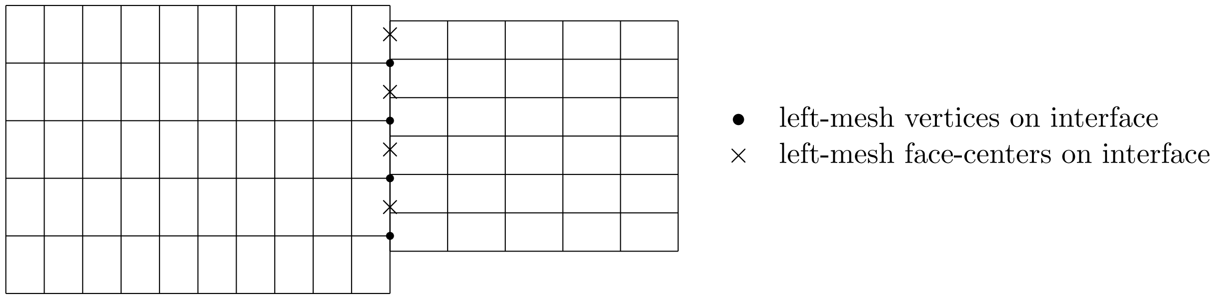
● left-mesh vertices on interface
× left-mesh face-centers on interface

Figure G.20 – A mismatched abutting interface

Each cell in an irregular mesh has at least one vertex in common with at least one other cell in the mesh. The connectivity and adjacency of the cells may be determined from the common vertices.

Each cell in an unstructured mesh is explicitly represented in terms of its shape and an ordered list of its vertices. The vertices are implied rather than being explicitly represented. Essentially all the vertices in a mesh can be mapped to a sequential list, and reference to a vertex is then equivalent to specifying the particular position in the list.

Figures G.19 to Figure G.21 show three types of mesh interfaces.

Figure G.19 illustrates a 1-to-1 abutting interface, also referred to as matching or C0 continuous. The interface is a plane of vertices that are physically coincident (i.e., they have identical coordinate values) between the adjacent meshes; grid-coordinate lines perpendicular to the interface are continuous from one mesh to the next. In 3-D, a 1-to-1 abutting interface is always a logically rectangular region.

The second type of interface, is mismatched abutting, where two meshes touch but do not overlap (except for vertices and cell faces on the grid plane of the interface). Vertices on the interface need not be physically concident between the two meshes. Figure G.20 indentifies the vertices and face centers of the left mesh that lie on the interface. In 3-D, the vertices of a mesh that constitute an interface patch may not form a logically rectangular region.

The third type of interface is called overset and occurs when two meshes overlap; in 3-D, the overlap is a 3-D region. For overset interfaces, one of the two meshes takes precedence over the other; this establishes which solution in the overlap region to retain and which one to discard. The region in a given mesh where the solution is discarded is called an overset hole and the mesh vertices outlining the hole are called fringe points.

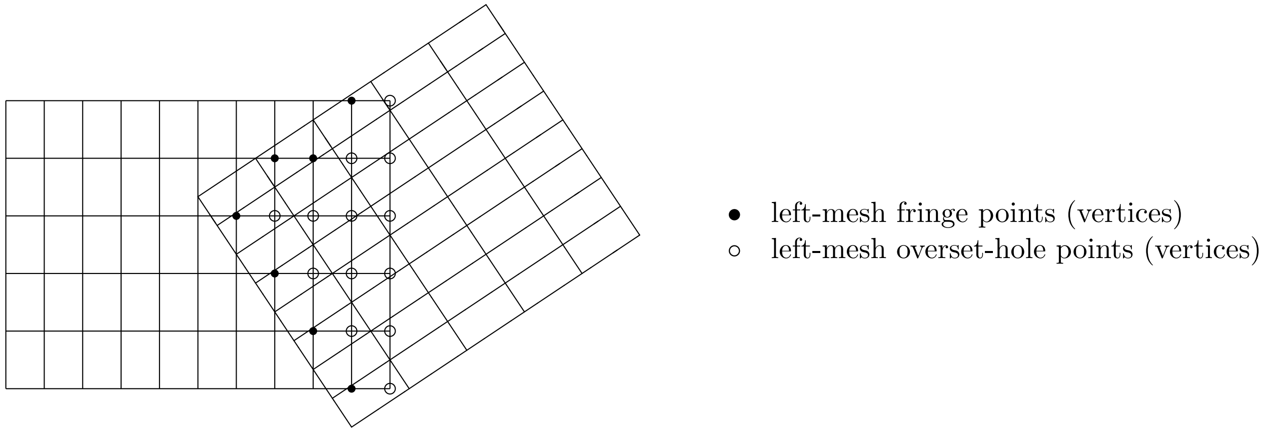
● left-mesh fringe points (vertices)
○ left-mesh overset-hole points (vertices)

Figure G.21 – An overset interface

Figure G.21 depicts an overlap region between two meshes, where the right mesh takes precedence over the left mesh. The points identified in Figure G.21 are the fringe points and overset-hole points for the left mesh. In addition, for the mesh taking precedence, any bounding points (i.e., vertices on the bounding faces) of the mesh that lies within the overlap must also be identified.

Overset interfaces may include multiple layers of fringe points outlining holes and at mesh boundaries.

For the mismatched abutting and overset interfaces in Figure G.20 and Figure G.21, the left mesh plays the role of receiver mesh and the right plays the role of donor mesh.

Mesh vertices that are included in a mesh interface are, in general, termed interface points.

G.3.2.2 Boundary-conditions

This model is an attempt to unify boundary-condition specifications within mesh-based numerical analysis codes. The structures and conventions presented are a compromise between simplicity and generality.

The difficulty with boundary-conditions is that there is such a wide variety used, and even a single boundary-condition equation is often implemented differently in different codes. Some boundary-conditions, such as a symmetry plane, are fairly well defined. Other boundary-conditions are much looser in their definition and implementation.

EXAMPLE 1 In CFD analyses an inflow boundary is an example of a loosely defined condition. It is generally accepted how many solution quantities should be specified at an inflow boundary (from mathematical well-posedness arguments), but what those quantities are will change with the class of flow problems (e.g., internal flows *vs.* external flows), and they will also change from code to code.

EXAMPLE 2 An additional difficulty for CFD analysis is that in some situations different boundary-condition equations are applied depending on local flow conditions. Any boundary where the flow can change from inflow to outflow or supersonic to subsonic is a candidate for flow-dependent boundary-condition equations.

These difficulties have moulded the design of the boundary-condition structures and conventions. Boundary-condition types are defined (**bc_type_simple**, **bc_type_compound**, and **bc_type**)

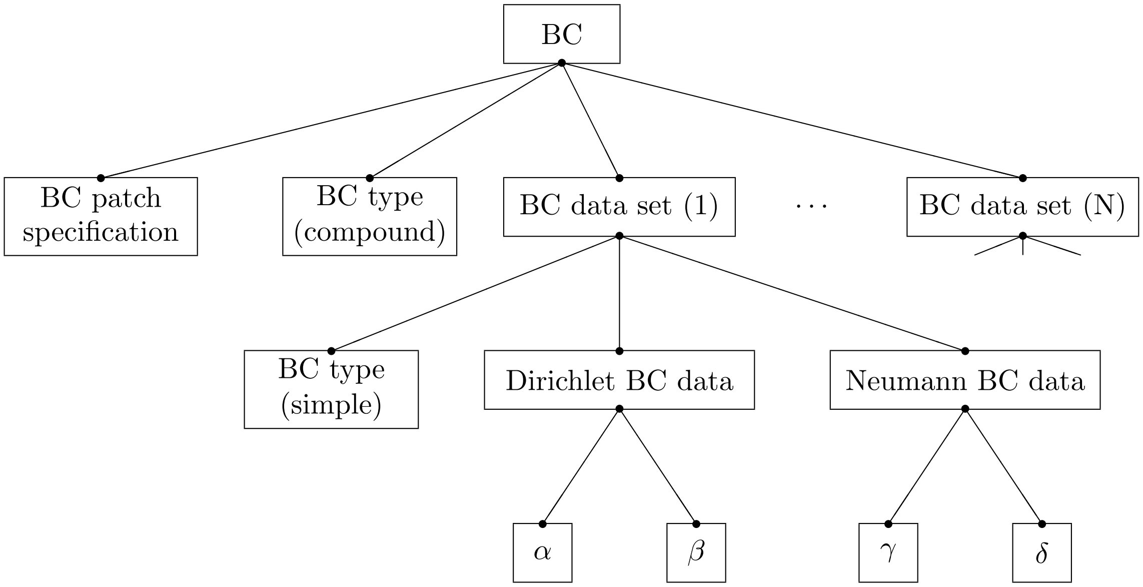

Figure G.22 – Hierarchy for boundary-condition structures

that establish the equations to be enforced. However, for those more loosely defined boundary-conditions, such as inflow/outlow, the boundary-condition type merely establishes general guidelines on the equations to be imposed. Augmenting (and superseding) the information provided by the boundary-condition type is precisely defined boundary-condition solution data. Data-name conventions are used to identify the exact quantities involved in the boundary-conditions.

One flexibility that is provided by this approach is that boundary-condition information can easily be built during the course of analysis. For example, during grid-generation phases minimal information (e.g., only the boundary-condition type) may be given. Then, prior to running of the solver, more specific boundary-condition information, such as Dirichlet or Neumann data, may be added.

Boundary-conditions are classified as either fixed or dependent. Fixed boundary-conditions enforce a given set of boundary-condition equations regardless of conditions; dependent boundary-conditions enforce different sets of boundary-condition equations depending on local conditions. Both fixed and dependent boundary-conditions are incorporated into a uniform framework, which allows all boundary-conditions to be described in a similar manner.

Figure G.22 depicts the hierarchy used for prescribing a single boundary-condition. Each boundary-condition includes a type that describes the general equations to enforce, a patch specification, and a collection of data sets. The minimum required information for any boundary-condition is the patch specification and the boundary-condition type. This minimum information is similar to that used in many existing (flow) solvers.

Generality in prescribing equations to enforce and their associated boundary-condition data is provided in the optional data sets. Each data set contains all boundary-condition data required for a given fixed or simple boundary-condition. Each data set is also tagged with a boundary-condition type. For fixed boundary-conditions, the hierarchical tree contains a single data set, and the two boundary-condition types shown in Figure G.22 are identical. Dependent or compound boundary-conditions contain multiple data sets, each to be applied separately depending on local conditions. The compound boundary-condition type describes the general dependent boundary-conditions, and each data set contains associated simple boundary-condition types.

EXAMPLE 3 In CFD analyses a farfield boundary condition would contain four data sets, where each applies to the different combinations of subsonic and supersonic inflow and outflow.

Within a single data set, boundary-condition data is grouped by equation type into Dirichlet and Neumann data. The lower leaves of Figure G.22 show data for generic solution quantities α and β to be applied in Dirichlet conditions, and data for γ and δ to be applied in Neumann boundary-conditions. **data_array** entities are employed to store these data and to identify the specific variables they are associated with.

In situations where the data sets (or any information contained therein) are absent from a given boundary-condition hierarchy, solvers are free to impose any appropriate boundary-conditions. Although not pictured in Figure G.22, it is also possible to specify the reference state from which the solver should extract the boundary-condition data.

G.3.2.3 Equations

The description of the equation set includes the general class of governing equations and mathematical models. Included with each equation description are associated constants.

The description of the flow-equation set includes the general class of governing equations, the turbulent closure equations, the gas model, and the viscosity and thermal-conductivity models. Included with each equation description are associated constants.

The structure definitions attempt to balance the opposing requirements of initial ease of implementation against requirements for extensibilty and future growth.

The intended use of these structures is primarily for archival purposes and to provide additional documentation of the solution.

NOTE 1 As an exampler, one small set of data name identifiers is specified for force and moment data which may be common to both structural and CFD analyses. It is expected that this set will be extended, and other sets introduced, for specific kinds of analyses.

NOTE 2 As an exampler, one small model for relating thermal properties is specified, which may cover both thermal and CFD analyses. It is expected that this model may have to be extended, and other kinds of model introduced, for specific kinds of analyses.

G.3.2.4 Results

The principal result of an analysis is the solution data over the computational grid.

Other data, such as equation residuals, can also form part of the results.

G.3.2.5 Numerical data

One practical aspect of some kinds of numerical analyses is that they involve massive amounts of numerical data. For example in CFD analyses the data includes coordinate values of meshes and the calculated results at each cell or node in the mesh; the numerical data may be Gigabytes in size.

The structure type **data_array** is a general purpose structure that may be used for holding data arrays and scalars. In the context of mesh-based numerical analysis it is used to describe grid coordinates, solution data, governing parameters, boundary-condition data, and other information. Information such as the number of dimensions in a multidimensional arry, the array sizes for each dimension, the dimensional units (if any), may be maintained by the **data_array**.

Several classes of data need to be addressed with the **data_array** structure type, for example:

a) dimensional data (e.g., velocity in units of m/s);

b) nondimensional data normalized by dimensional reference quantities;

c) nondimensional data with associated nondimensional reference quantities;

d) nondimensional parameters (e.g., Reynolds number, pressure coefficient);

e) pure constants (e.g., π, e).

Each of the classes of data requires different information to describe dimensional units or normalization associated with the data.

Identifiers or data-names need to be associated with instances of **data_array** entities to identify and describe the quantity being stored; as examples, one instance may be holding data related to coordinate values while another holds data related to density values. To facilitate communication between different application codes, an initial set of standardized data-name identifiers with fairly precise definitions are provided. For any identifier in this set, the associated data should be unambiguously understood. In essence, this schema supplies standardized terminology for labeling analysis-related data such as grid coordinates.

All standardized identifiers denote scalar quantities; this is consistent with the intended use of the **data_array** structure type to describe an array of scalars. For quantities that are vectors, such as velocity, their components are listed.

Included with the lists of standard data-name identifiers, the fundamental units of dimensions associated with that quantity are provided. The following notation is used for the fundamental units: **M** is mass, **L** is length, **T** is time, Θ is temperature and α is angle. These fundamental units are directly associated with the elements of the **dimensional_exponents** structure. For example, a quantity that has dimensions **ML/T** corresponds to **mass_exponent** = +1, **length_exponent** = +1, and **time_exponent** = -1.

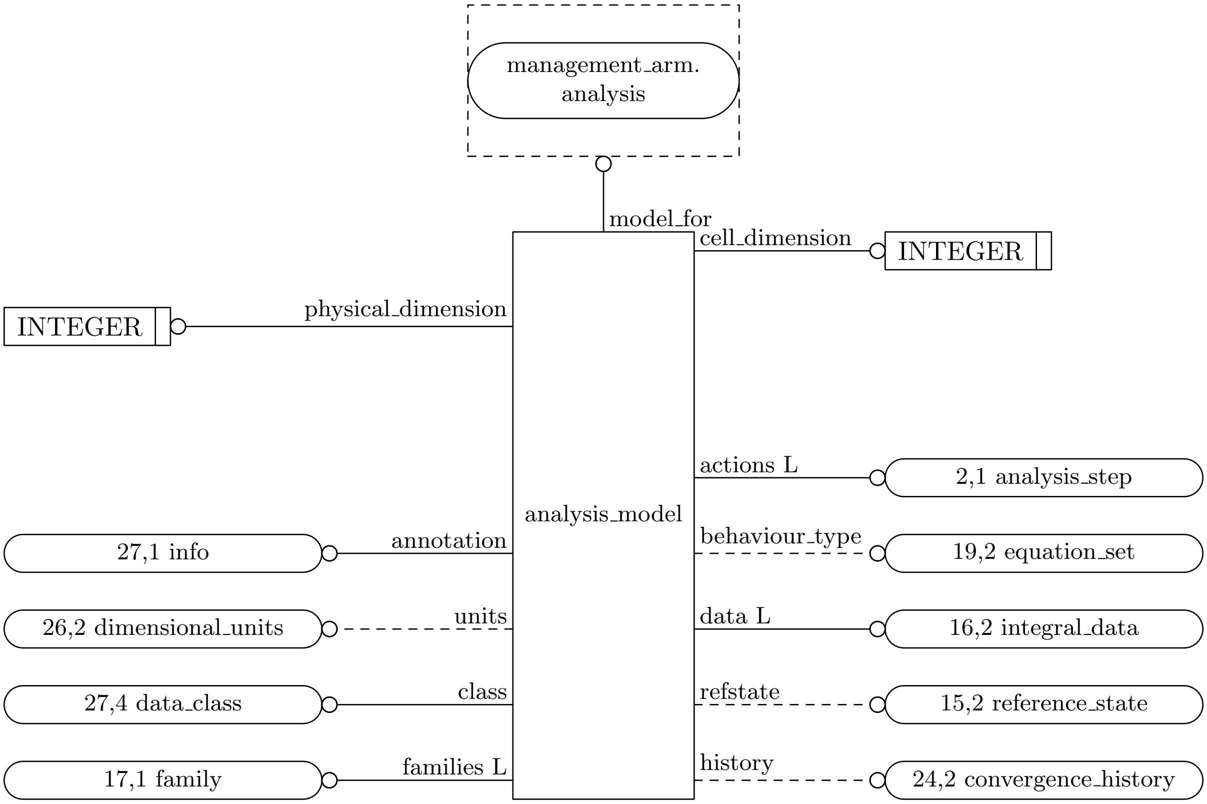

Figure G.23 – Partial entity level diagram of ARM analysis_arm schema (page 1 of 28)

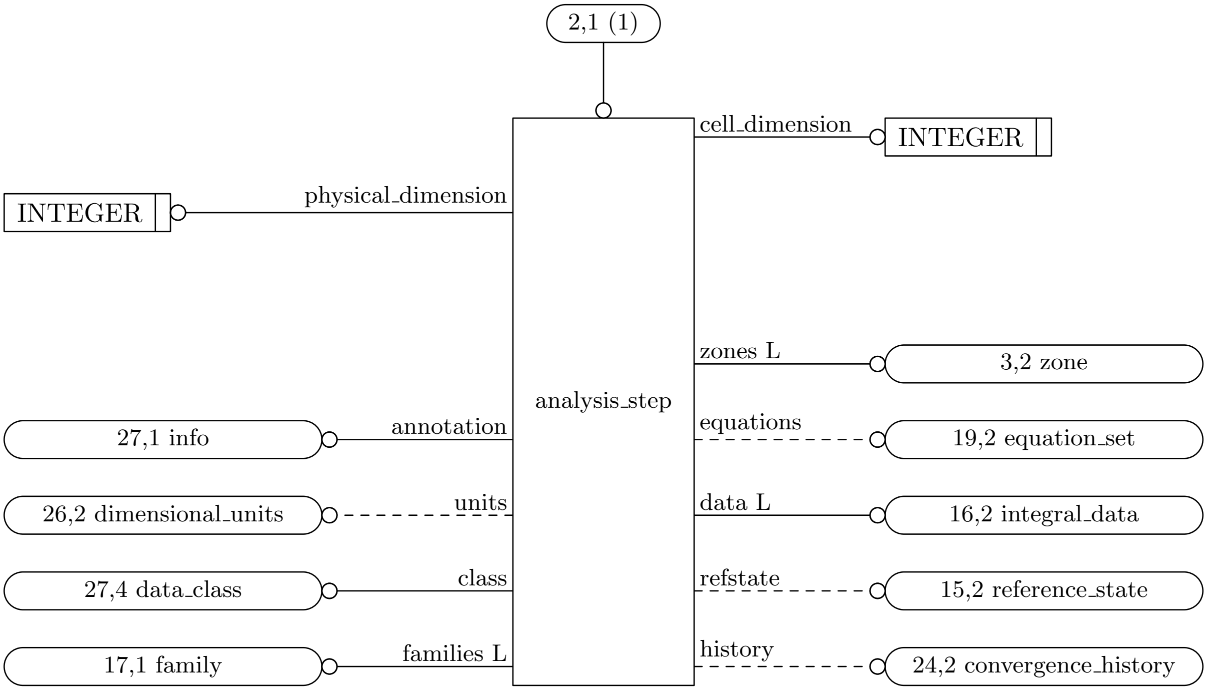

Figure G.24 – Partial entity level diagram of ARM analysis_arm schema (page 2 of 28)

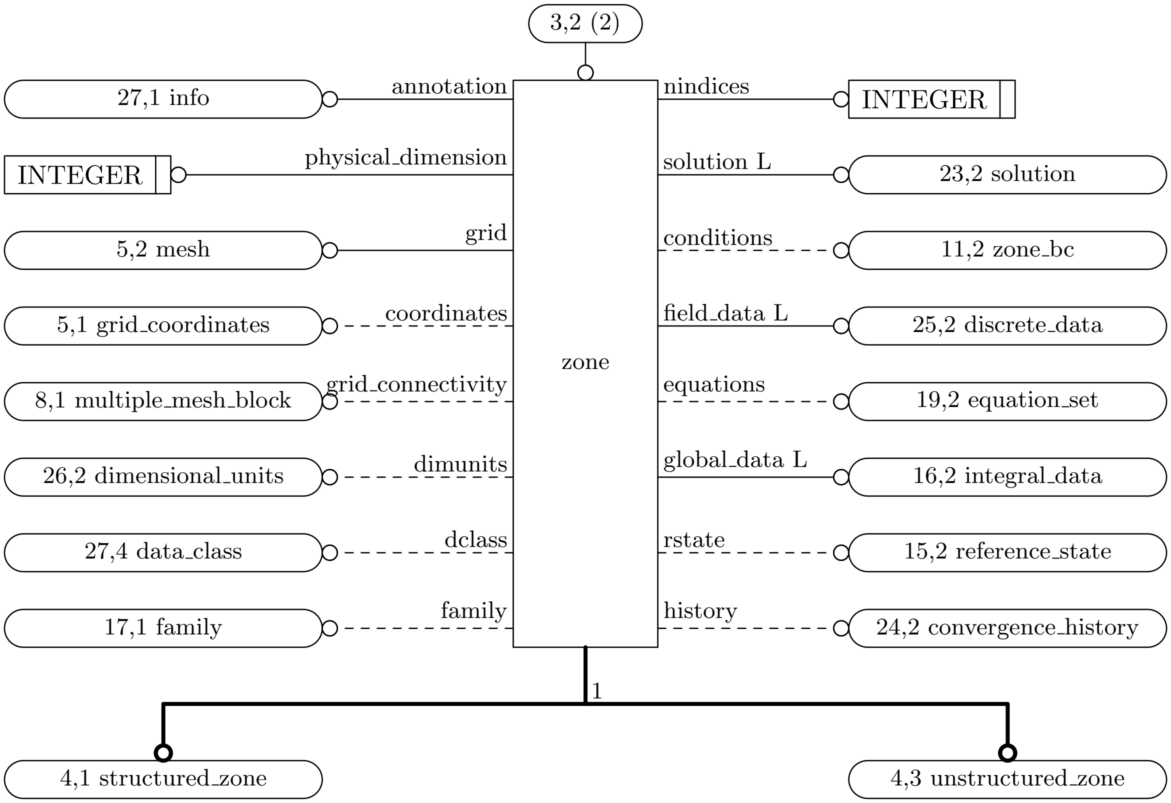

Figure G.25 – Partial entity level diagram of ARM analysis_arm schema (page 3 of 28)

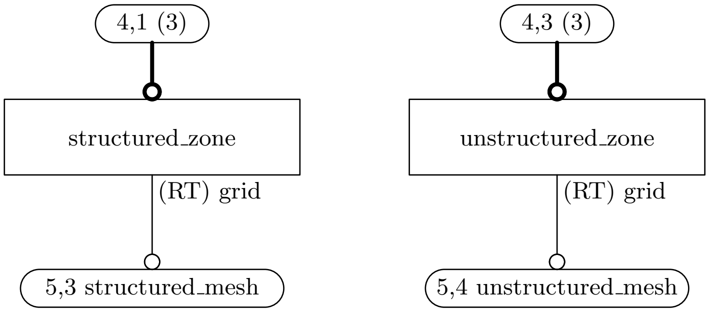

Figure G.26 – Partial entity level diagram of ARM analysis_arm schema (page 4 of 28)

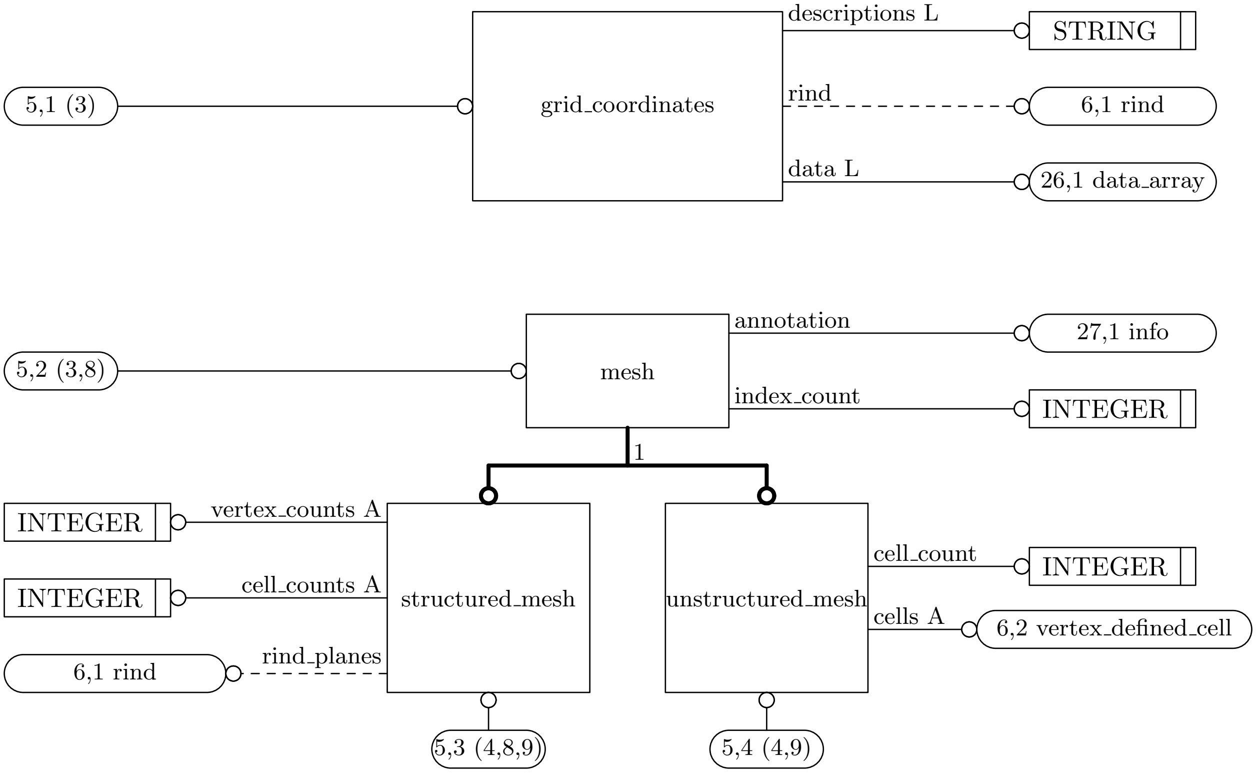

Figure G.27 – Partial entity level diagram of ARM analysis_arm schema (page 5 of 28)

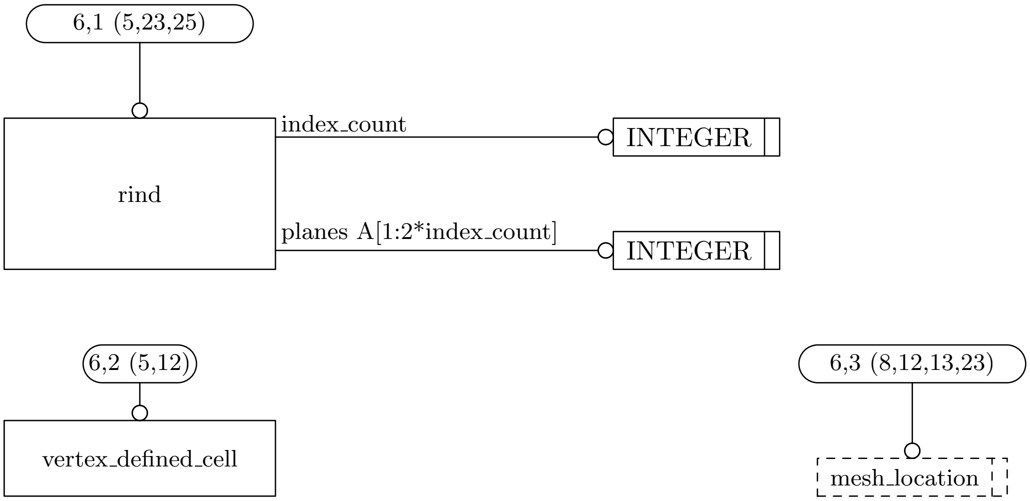

Figure G.28 – Partial entity level diagram of ARM analysis_arm schema (page 6 of 28)

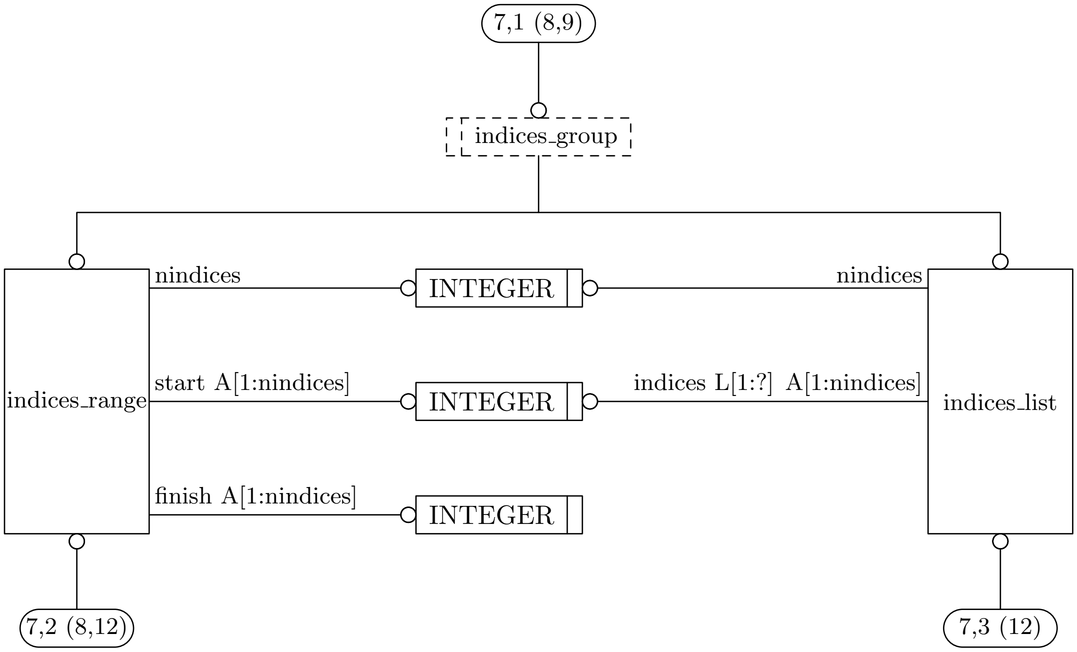

Figure G.29 – Partial entity level diagram of ARM analysis_arm schema (page 7 of 28)

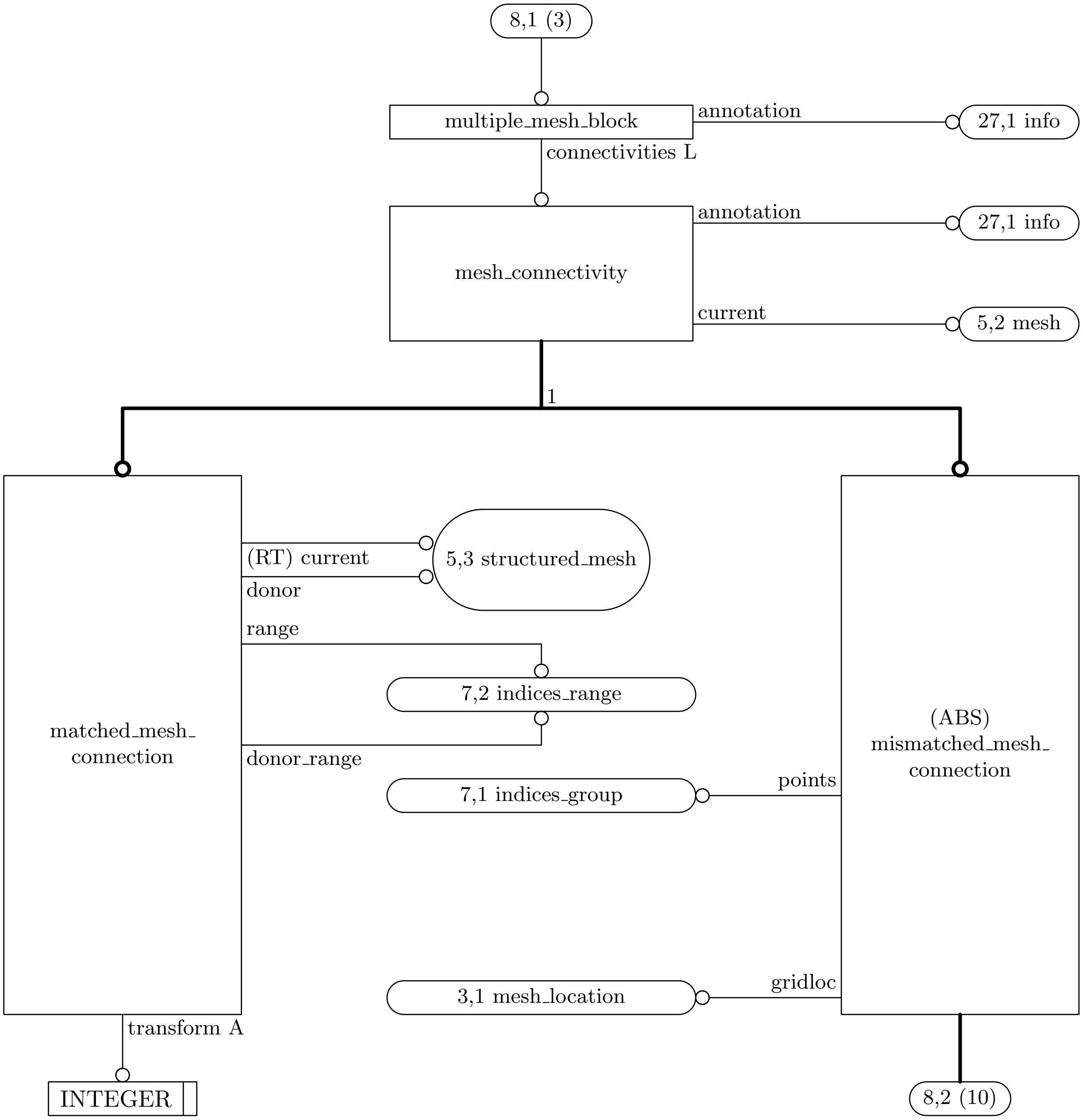

Figure G.30 – Partial entity level diagram of ARM analysis_arm schema (page 8 of 28)

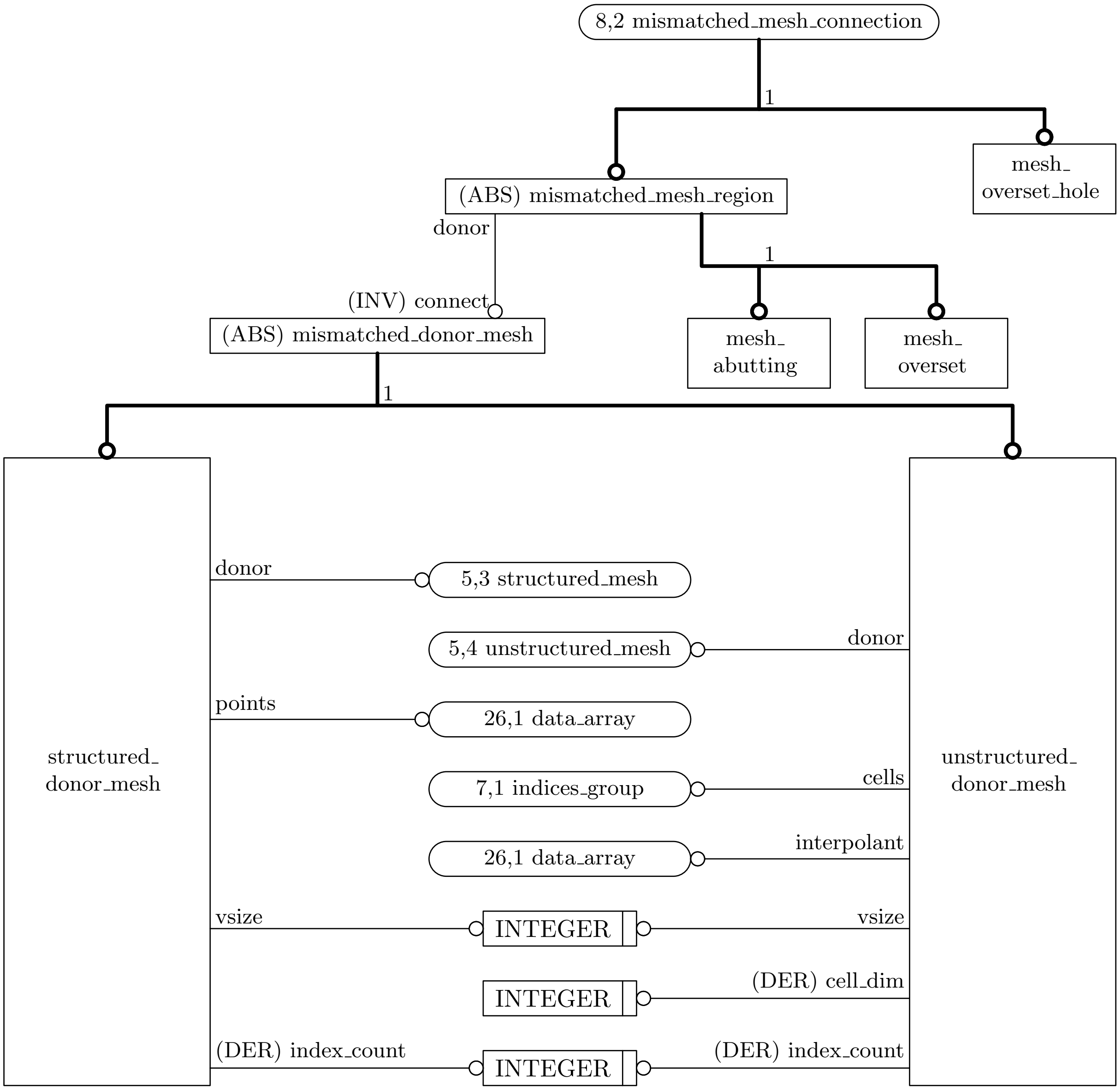

Figure G.31 – Partial entity level diagram of ARM analysis_arm schema (page 9 of 28)

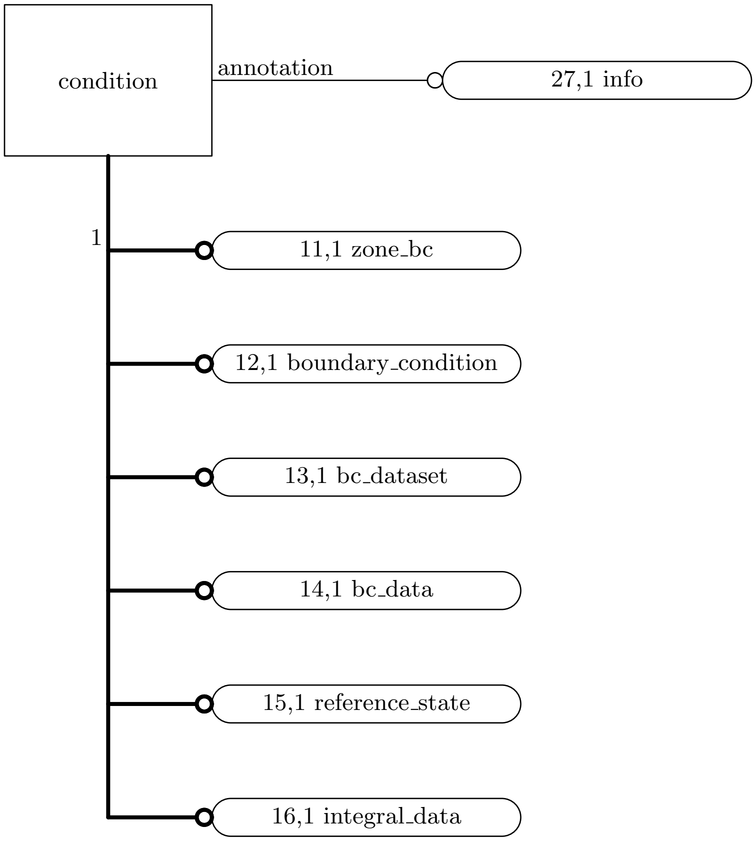

Figure G.32 – Partial entity level diagram of ARM analysis_arm schema (page 10 of 28)

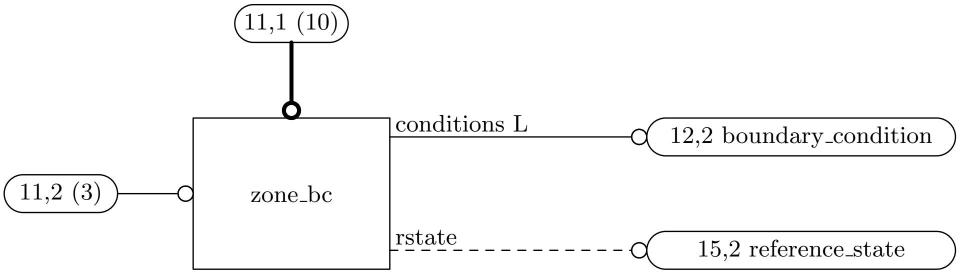

Figure G.33 – Partial entity level diagram of ARM analysis_arm schema (page 11 of 28)

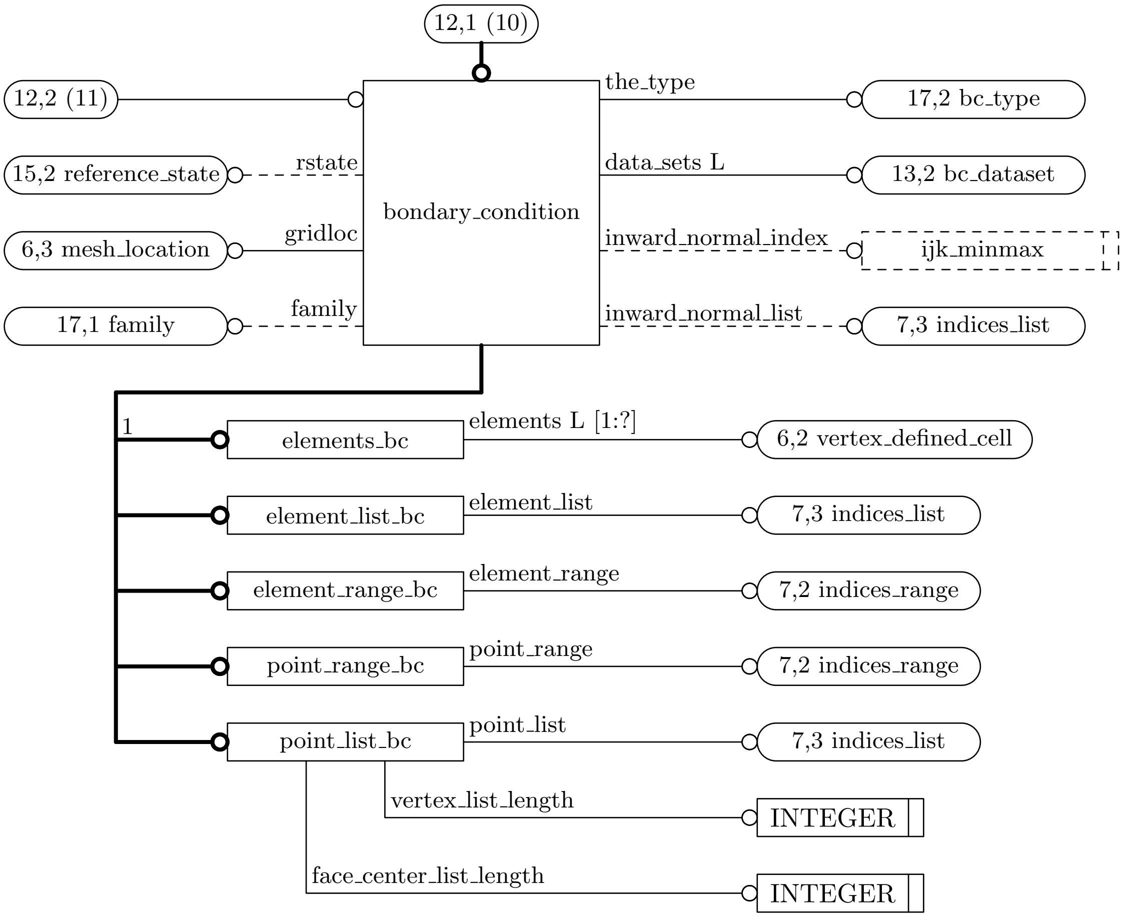

Figure G.34 – Partial entity level diagram of ARM analysis_arm schema (page 12 of 28)

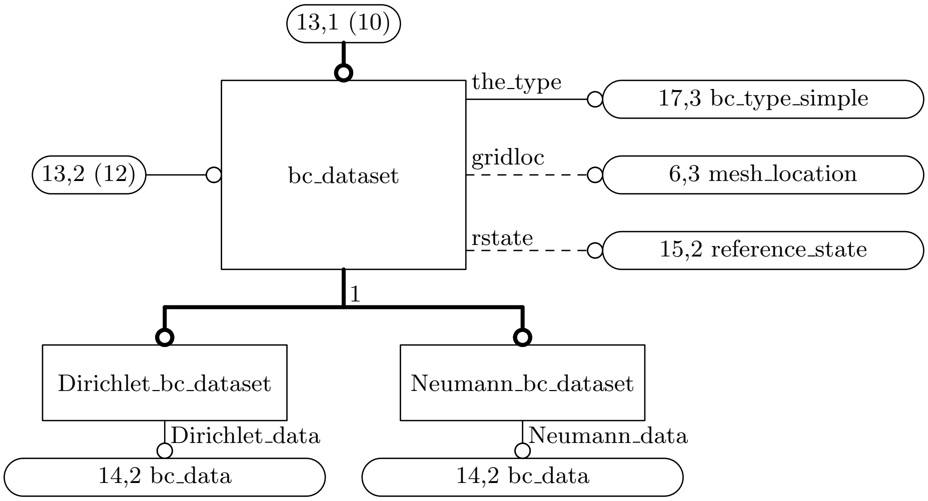

Figure G.35 – Partial entity level diagram of ARM analysis_arm schema (page 13 of 28)

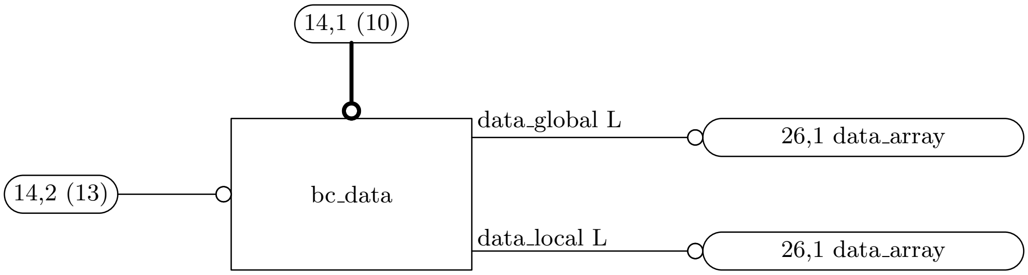

Figure G.36 – Partial entity level diagram of ARM analysis_arm schema (page 14 of 28)

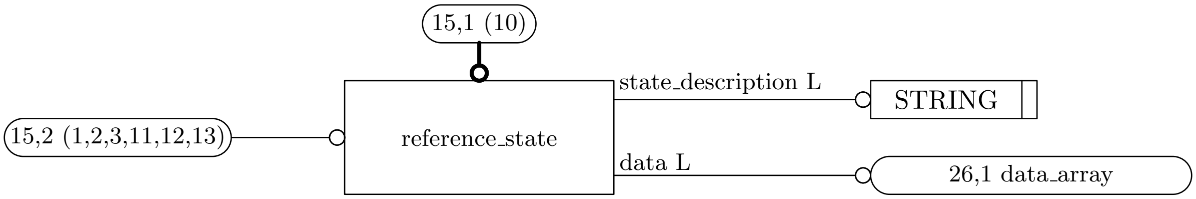

Figure G.37 – Partial entity level diagram of ARM analysis_arm schema (page 15 of 28)

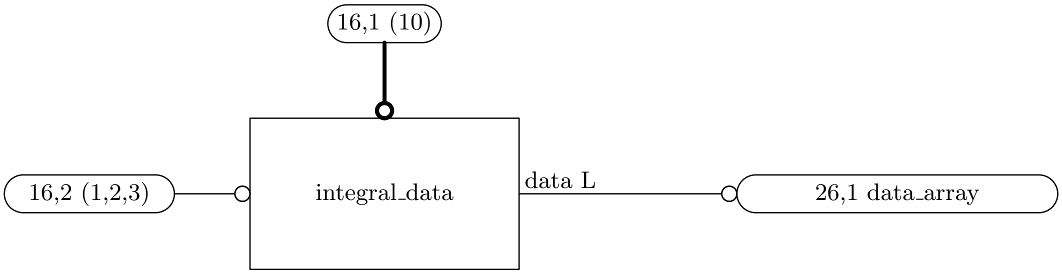

Figure G.38 – Partial entity level diagram of ARM analysis_arm schema (page 16 of 28)

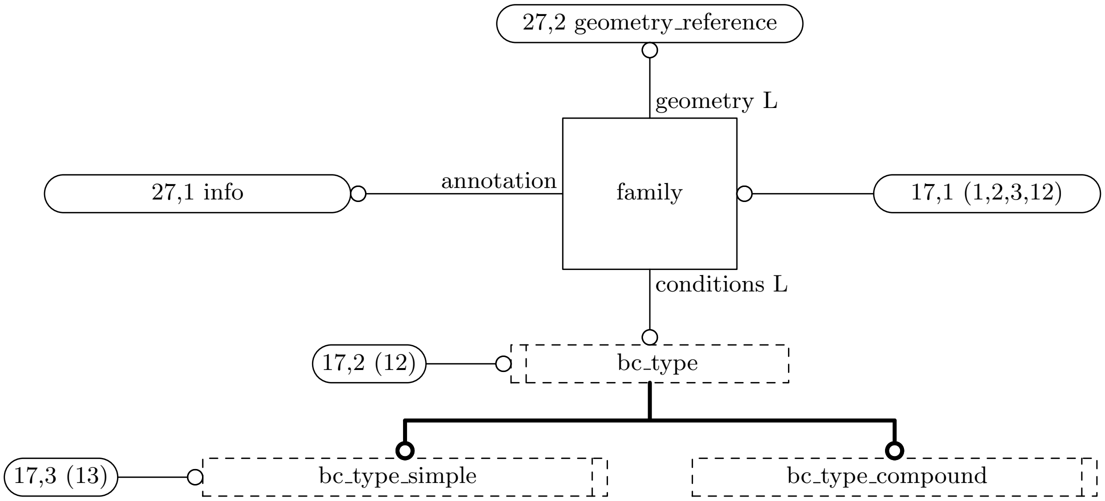

Figure G.39 – Partial entity level diagram of ARM analysis_arm schema (page 17 of 28)

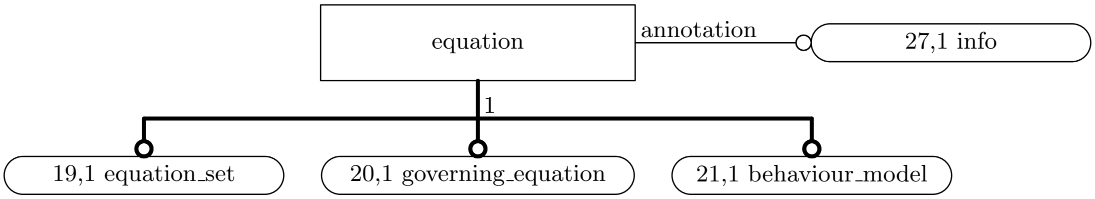

Figure G.40 – Partial entity level diagram of ARM analysis_arm schema (page 18 of 28)

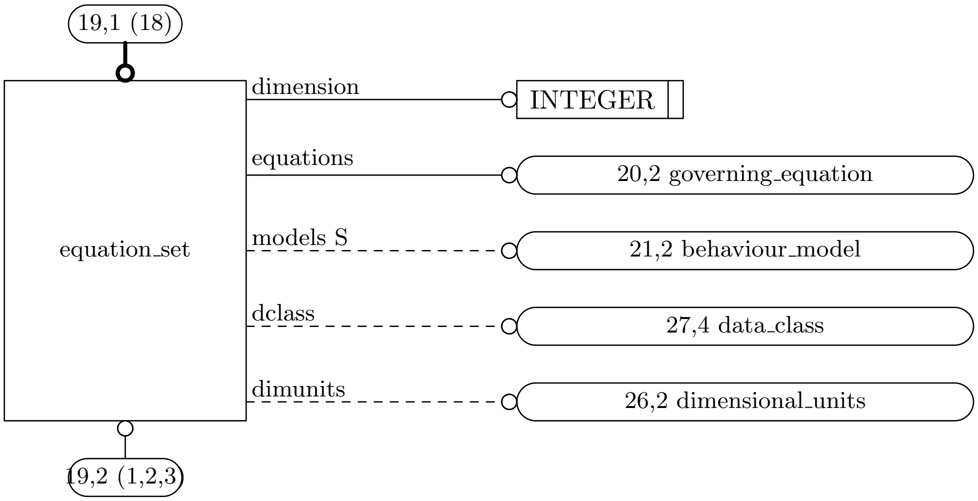

Figure G.41 – Partial entity level diagram of ARM analysis_arm schema (page 19 of 28)

G.3.3 analysis_arm type definitions

G.3.3.1 indices_group

indices_group is a selection of a group of indices into a multi-dimensional array.

EXPRESS specification:

```
*)
TYPE indices_group = SELECT
     (indices_list,
      indices_range);
END_TYPE;
(*
```

G.3.3.2 mesh_location

mesh_location is an enumeration of locations with respect to a mesh.

EXPRESS specification:

```
*)
```

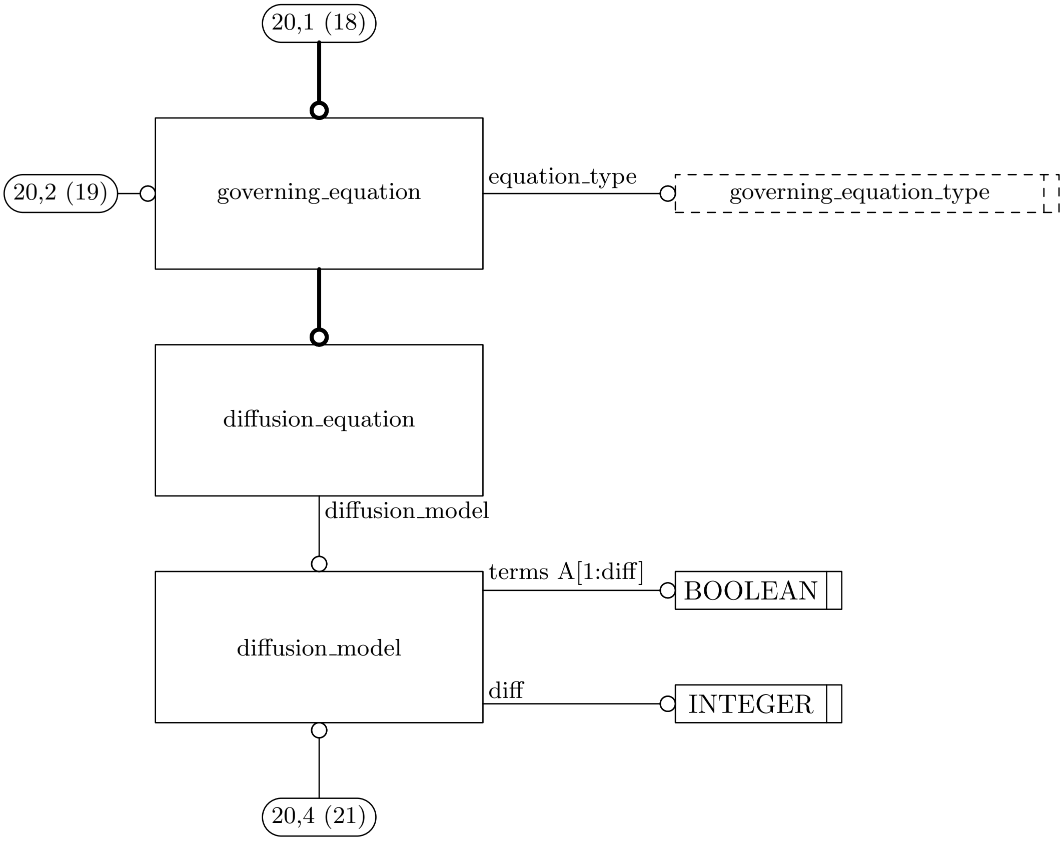

Figure G.42 – Partial entity level diagram of ARM analysis_arm schema (page 20 of 28)

```
TYPE mesh_location = EXTENSIBLE ENUMERATION OF
     (unspecified,
      application_defined,
      vertices,
      cell_center,
      face_center,
      iface_center,
      jface_center,
      kface_center,
      edge_center);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: is unspecified;

application_defined: is specified via an external agreement between the data creator and the data user;

vertices: is coincident with the mesh vertices;

cell_center: is the center of a cell; this is also appropriate for entities associated with cells but not necessarily with a given location in a cell;

face_center: is the center of a generic face which can point in any coordinate direction;

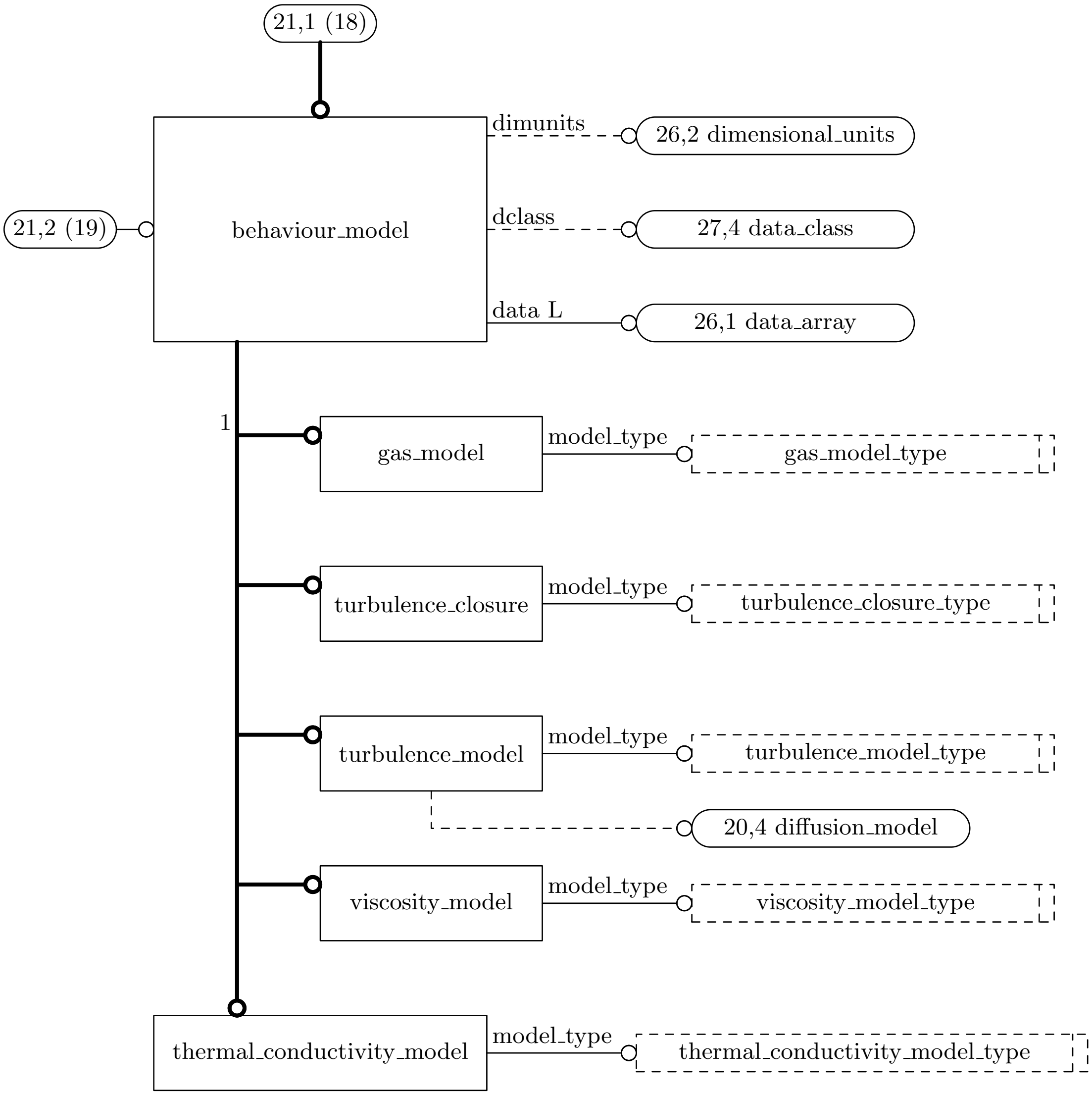

Figure G.43 – Partial entity level diagram of ARM analysis_arm schema (page 21 of 28)

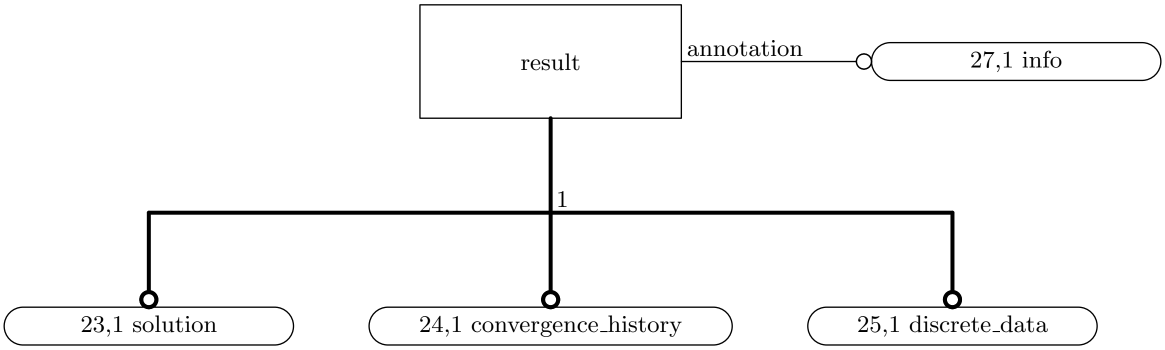

Figure G.44 – Partial entity level diagram of ARM analysis_arm schema (page 22 of 28)

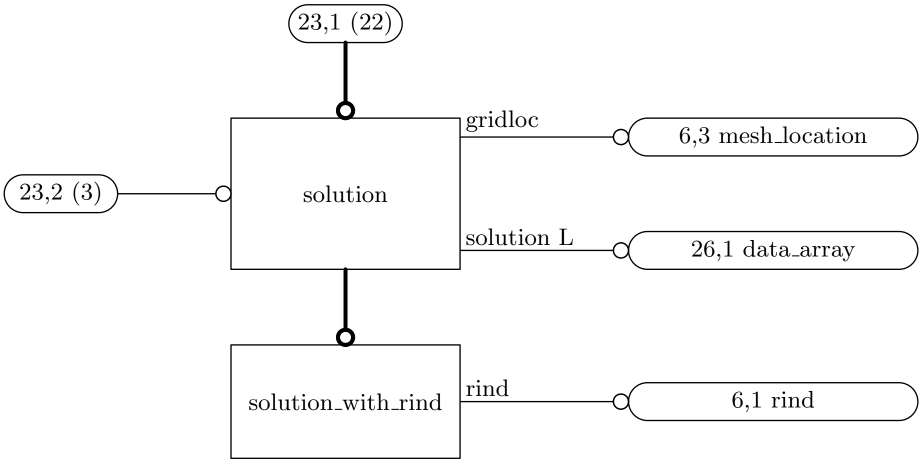

Figure G.45 – Partial entity level diagram of ARM analysis_arm schema (page 23 of 28)

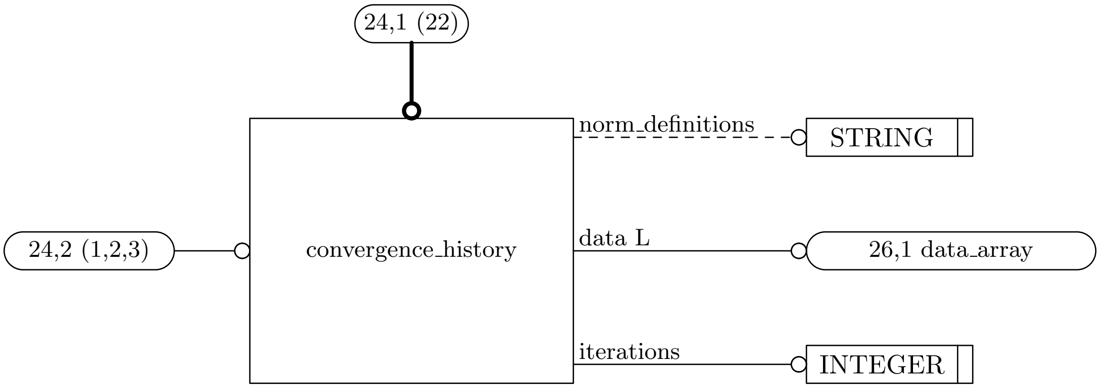

Figure G.46 – Partial entity level diagram of ARM analysis_arm schema (page 24 of 28)

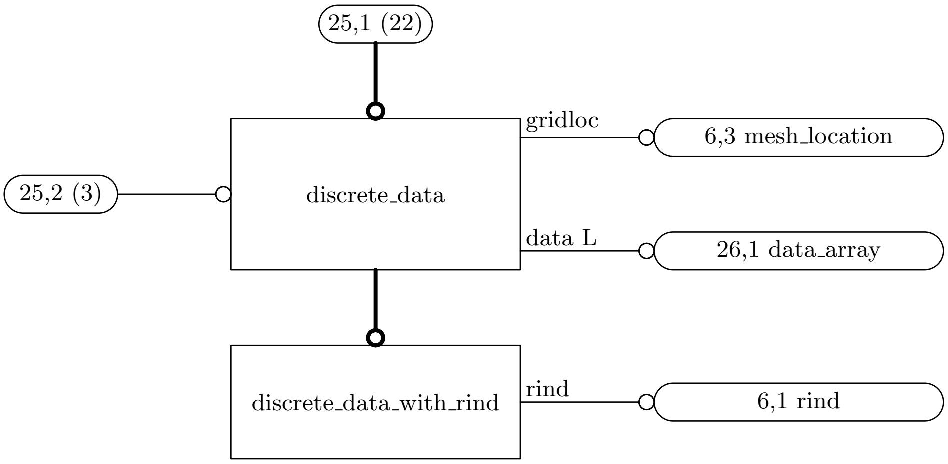

Figure G.47 – Partial entity level diagram of ARM analysis_arm schema (page 25 of 28)

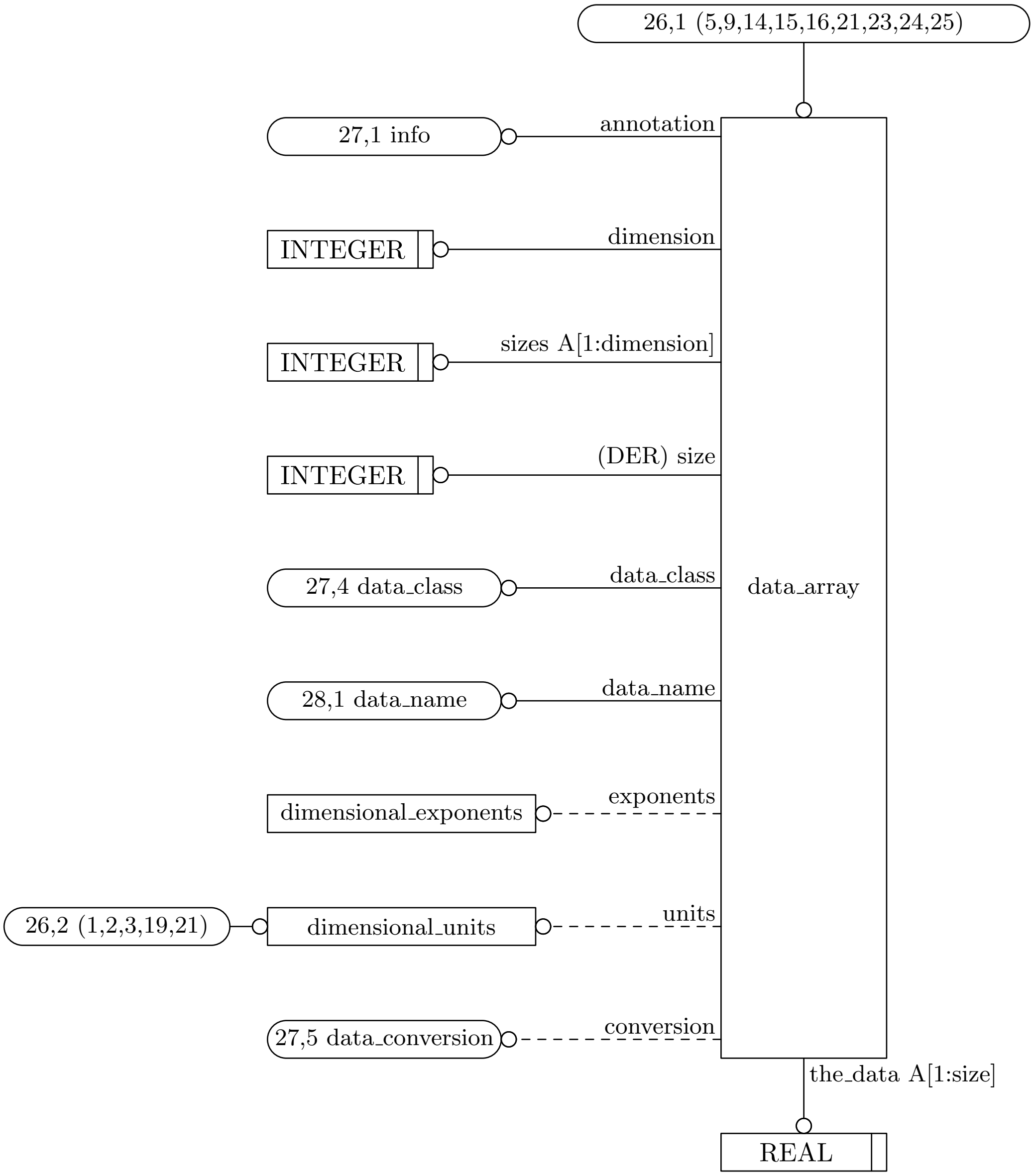

Figure G.48 – Partial entity level diagram of ARM analysis_arm schema (page 26 of 28)

iface_center: is the center of a face in 3-D whose computational normal points in the i direction;

jface_center: is the center of a face in 3-D whose computational normal points in the j direction;

kface_center: is the center of a face in 3-D whose computational normal points in the k direction;

edge_center: is the center of an edge.

G.3.3.3 data_name

data_name is a defined identifier for data names.

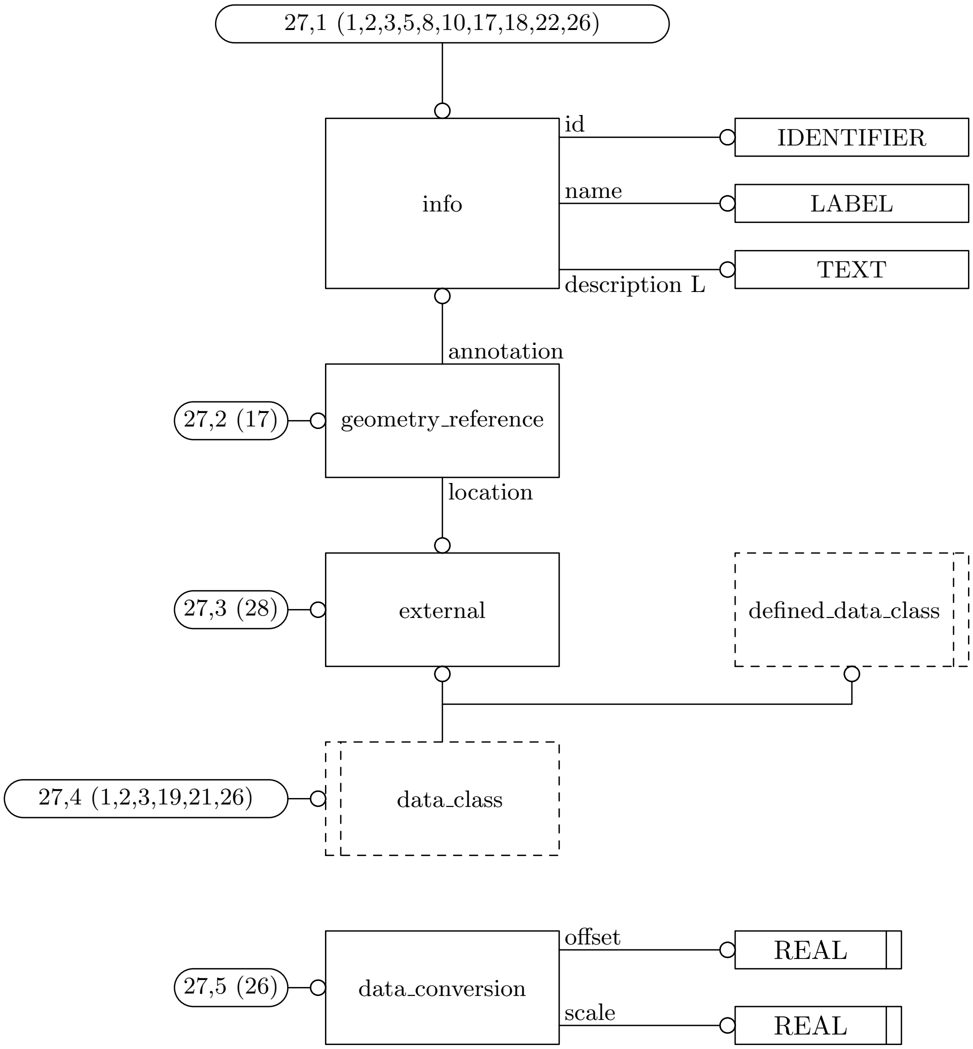

Figure G.49 – Partial entity level diagram of ARM analysis_arm schema (page 27 of 28)

EXPRESS specification:

```
*)
TYPE data_name = SELECT
     (defined_data_name,
      external);
END_TYPE;
-- map to choose_general_property_identifier (part 110)
(*
```

G.3.3.4 defined_data_name

defined_data_name is a kind of defined identifier for data names.

EXPRESS specification:

```
*)
```

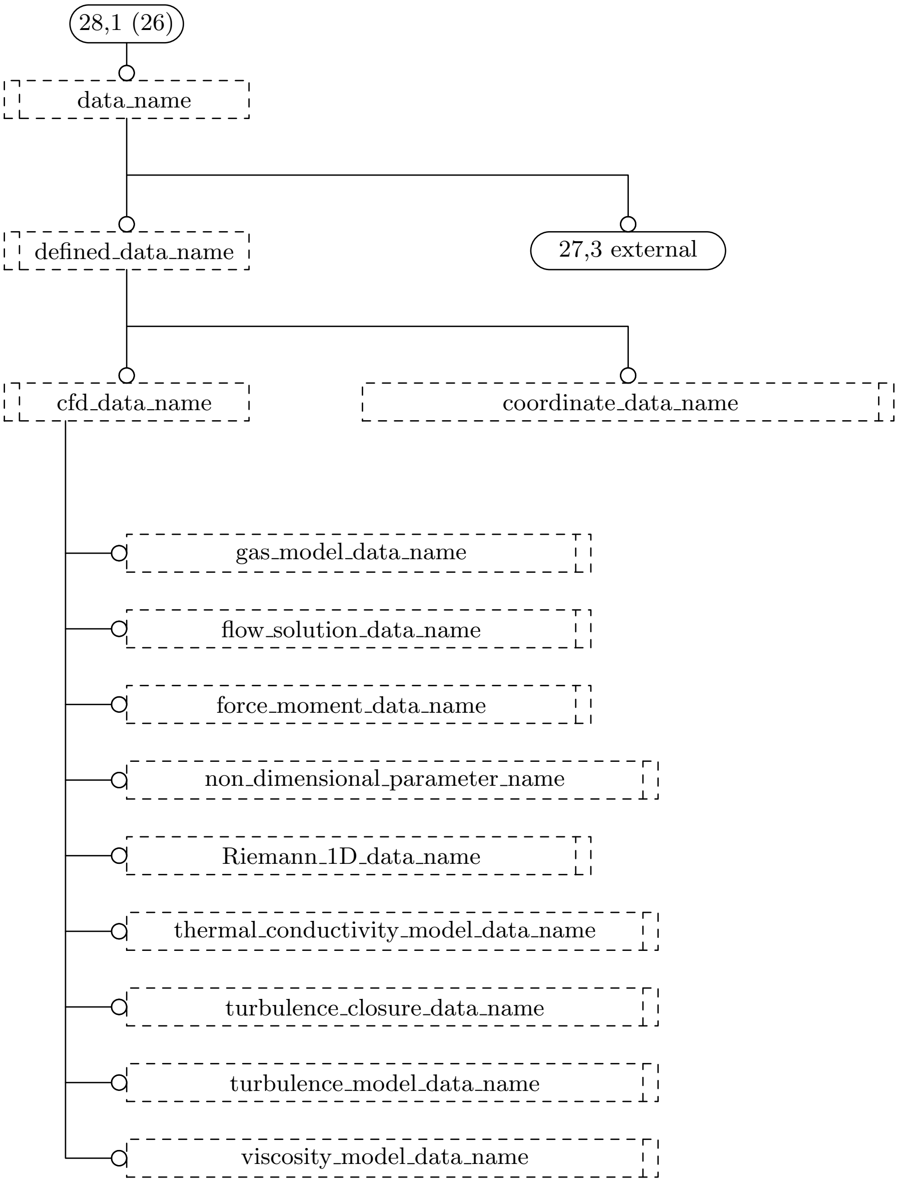

Figure G.50 – Partial entity level diagram of ARM analysis_arm schema (page 28 of 28)

```
TYPE defined_data_name = SELECT
     (coordinate_data_name,
      cfd_data_name);
END_TYPE;
(*
```

G.3.3.5 cfd_data_name

cfd_data_name is a kind of defined identifier for CFD data names.

EXPRESS specification:

```
*)
TYPE cfd_data_name = SELECT
```

```
      (nondimensional_parameter_name,
       Riemann_1D_data_name,
       force_moment_data_name,
       gas_model_data_name,
       viscosity_model_data_name,
       thermal_conductivity_model_data_name,
       turbulence_closure_data_name,
       turbulence_model_data_name,
       flow_solution_data_name);
END_TYPE;
(*
```

G.3.3.6 nondimensional_parameter_name

nondimensional_parameter_name is an enumeration of standardized nondimensional parameter data names.

CFD codes are rich in nondimensional governing parameters, such as Mach number and Reynolds number, and nondimensional flowfield coefficients, such as pressure coefficient. The problem with these parameters is that their definitions and conditions that they are evaluated at can vary from code to code. Reynolds number is particularly notorious in this respect.

These parameters have posed us with a difficult dilemma. Either we impose a rigid definition for each and force all database users to abide by it, or we develop some methodology for describing the particular definition that the user is employing. The first limits applicability and flexibility, and the second adds complexity. We have opted for the second approach, but we include only enough information about the definition of each parameter to allow for conversion operations. For example, the Reynolds number includes velocity, length and kinematic viscosity scales in its definition (i.e. $Re = VL/\nu$). The database description of Reynolds number includes these different scales. By providing these 'definition components', any code that reads Reynolds number from the database can transform its value to an appropriate internal definition.

Definitions for nondimensional flowfield coefficients follow: The pressure coefficient is defined as,

$$c_p = \frac{p - p_{\mathrm{ref}}}{\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2},$$

where $\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2$ is the dynamic pressure evaluated at some reference condition, and p_{ref} is some reference pressure. The skin friction coefficient is,

$$\vec{c}_f = \frac{\vec{\tau}}{\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2},$$

where $\vec{\tau}$ is the shear stress or skin friction vector. Usually, $\vec{\tau}$ is evaluated at the wall surface.

EXPRESS specification:

```
*)
TYPE nondimensional_parameter_name = ENUMERATION OF
      (Mach,
       Mach_velocity,
       Mach_velocity_sound,
       Reynolds,
       Reynolds_velocity,
```

Table G.1 – Nondimensional parameter name identifiers

Data name identifier	**Description**	**Units**
Mach	Mach number: $M = q/c$	-
Mach_velocity	velocity scale (q)	$\mathbf{L}/\mathbf{T}$
Mach_velocity_sound	speed of sound scale (c)	$\mathbf{L}/\mathbf{T}$
Reynolds	Reynolds number: $Re = VL/\nu$	-
Reynolds_velocity	velocity scale (V)	$\mathbf{L}/\mathbf{T}$
Reynolds_length	length scale (L)	$\mathbf{L}$
Reynolds_viscosity_kinematic	kinematic viscosity scale (ν)	$\mathbf{L}^2/\mathbf{T}$
Prandtl	Prandtl number: $Pr = \mu c_p/k$	-
Prandtl_thermal_conductivity	thermal conductivity scale (k)	$\mathbf{ML}/(\mathbf{T}^3\Theta)$
Prandtl_viscosity_molecular	molecular viscosity scale (μ)	$\mathbf{M}/(\mathbf{LT})$
Prandtl_specific_heat_pressure	specific heat scale (c_p)	$\mathbf{L}^2/(\mathbf{T}^2\Theta)$
specific_heat_ratio	specific heat ratio: $\gamma = c_p/c_v$	-
specific_heat_ratio_pressure	specific heat at constant pressure (c_p)	$\mathbf{L}^2/(\mathbf{T}^2\Theta)$
specific_heat_ratio_volume	specific heat at constant volume (c_v)	$\mathbf{L}^2/(\mathbf{T}^2\Theta)$
coef_pressure	c_p	-
coef_pressure_dynamic	$1/2\rho_{\mathrm{ref}}q_{\mathrm{ref}}^2$	$\mathbf{M}/(\mathbf{LT}^2)$
coef_pressure_reference	p_{ref}	$\mathbf{M}/(\mathbf{LT}^2)$
coef_skin_friction_x	$\vec{c}_f \cdot \hat{e}_x$	-
coef_skin_friction_y	$\vec{c}_f \cdot \hat{e}_y$	-
coef_skin_friction_z	$\vec{c}_f \cdot \hat{e}_z$	-
length_reference	L_{ref}	$\mathbf{L}$

```
      Reynolds_length,
      Reynolds_viscosity_kinematic,
      Prandtl,
      Prandtl_thermal_conductivity,
      Prandtl_viscosity_molecular,
      Prandtl_specific_heat_pressure,
      specific_heat_ratio,
      specific_heat_ratio_pressure,
      specific_heat_ratio_volume,
      coef_pressure,
      coef_skin_friction_x,
      coef_skin_friction_y,
      coef_skin_friction_z,
      coef_pressure_dynamic,
      coef_pressure_reference,
      length_reference);
END_TYPE;
(*
```

The meanings of the identifiers are given in Table G.1.

G.3.3.7 data_class

data_class is an identifier for classes of data.

EXPRESS specification:

```
*)
TYPE data_class = SELECT
     (defined_data_class,
      external);
END_TYPE;
-- map to choose_general_property_identifier (part 110)
(*
```

G.3.3.8 defined_data_class

defined_data_class is a kind of defined identifier for classes of data.

EXPRESS specification:

```
*)
TYPE defined_data_class = ENUMERATION OF
     (unspecified,
      application_defined,
      dimensional,
      normalised_by_dimensional,
      normalise_by_unknown_dimensional,
      dimensionless_parameter,
      dimensionless_constant);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: no identification;

application_defined: meaning is assigned by agreement external to this standard;

dimensional: identifies dimensional data;

normalised_by_dimensional: identifies nondimensional data that is normalised by dimensional reference quantities;

normalised_by_unknown_dimensional: identifies nondimensional data typically found in a completely nondimensional object base where all field and reference data is nondimensional;

nondimensional_parameter: identifies nondimensional parameters;

dimensionless_constant: identifies a constant value.

G.3.3.9 coordinate_data_name

coordinate_data_name is an enumeration of standardized coordinate systems data.

Coordinate systems for identifying physical location are as follows:

Table G.2 – Coordinate data name identifiers

Data name identifier	Description	Units
coordinate_x	x	**L**
coordinate_y	y	**L**
coordinate_z	z	**L**
coordinate_r	r	**L**
coordinate_theta	θ	α
coordinate_phi	ϕ	α
coordinate_normal	coordinate in direction of $\hat{e}_n$	**L**
coordinate_tangential	tangential coordinate (2–D only)	**L**
coordinate_xi	ξ	**L**
coordinate_eta	η	**L**
coordinate_zeta	ζ	**L**
coordinate_transform	transformation matrix (**T**)	-

System	3–D	2–D
Cartesian	(x, y, z)	(x, y) or (x, z) or (y, z)
Cylindrical	(r, θ, z)	(r, θ)
Spherical	(r, θ, ϕ)	
Auxiliary	(ξ, η, ζ)	(ξ, η) or (ξ, ζ) or (η, ζ)

Associated with these coordinate systems are the following unit vector conventions:

x-direction	$\hat{e}_x$	r-direction	$\hat{e}_r$	ξ-direction	$\hat{e}_\xi$
y-direction	$\hat{e}_y$	θ-direction	$\hat{e}_\theta$	η-direction	$\hat{e}_\eta$
z-direction	$\hat{e}_z$	ϕ-direction	$\hat{e}_\phi$	ζ-direction	$\hat{e}_\zeta$

Note that $\hat{e}_r$, $\hat{e}_\theta$ and $\hat{e}_\phi$ are functions of position.

It is expected that one of the 'standard' coordinate systems (cartesian, cylindrical or spherical) will be used within a zone (or perhaps the entire database) to define grid coordinates and other related data. The auxiliary coordinates will be used for special quantities, including forces and moments, which may not be defined in the same coordinate system as the rest of the data. When auxiliary coordinates are used, a transformation must also be provided to uniquely define them.

EXAMPLE 1 The transform from cartesian to orthogonal auxiliary coordinates is,

$$\begin{pmatrix} \hat{e}_\xi \\ \hat{e}_\eta \\ \hat{e}_\zeta \end{pmatrix} = \mathbf{T} \begin{pmatrix} \hat{e}_x \\ \hat{e}_y \\ \hat{e}_z \end{pmatrix},$$

where **T** is an orthonormal matrix (2×2 in 2–D and 3×3 in 3–D).

In addition, normal and tangential coordinates are often used to define boundary conditions and data related to surfaces. The normal coordinate is identified as n with the unit vector $\hat{e}_n$.

EXPRESS specification:

```
*)
TYPE coordinate_data_name = ENUMERATION OF
     (coordinate_x,
      coordinate_y,
      coordinate_z,
      coordinate_r,
      coordinate_theta,
      coordinate_phi,
      coordinate_normal,
      coordinate_tangential,
      coordinate_xi,
      coordinate_eta,
      coordinate_zeta,
      coordinate_transform);
END_TYPE;
(*
```

The meanings of the enumerated coordinate data identifiers are given in Table G.2.

G.3.3.10 Riemann_1D_data_name

Riemann_1D_data_name is an enumeration of standardized Riemann data for 1-D flow.

Boundary condition specification for inflow/outflow or farfield boundaries often involves Riemann invariants or characteristics of the linearized inviscid flow equations. For an ideal compressible gas, these are typically defined as follows: Riemann invariants for an isentropic 1–D flow are,

$$\left[\frac{\partial}{\partial t} + (u \pm c)\frac{\partial}{\partial x}\right]\left(u \pm \frac{2}{\gamma - 1}c\right) = 0.$$

Characteristic variables for the 3–D Euler equations linearized about a constant mean flow are,

$$\left[\frac{\partial}{\partial t} + \bar{\Lambda}_n \frac{\partial}{\partial x}\right] W'_n(x,t) = 0, \qquad n = 1, 2, \ldots 5,$$

where the characteristics and corresponding characteristic variables are

Characteristic	$\bar{\Lambda}_n$	W'_n
‘entropy’	$\bar{u}$	$p' - \rho'/\bar{c}^2$
‘vorticity’	$\bar{u}$	v'
‘vorticity’	$\bar{u}$	w'
‘acoustic’	$\bar{u} \pm \bar{c}$	$p' \pm u'/(\bar{\rho}\bar{c})$

Barred quantities are evaluated at the mean flow, and primed quantities are linearized perturbations. The only non-zero mean-flow velocity component is $\bar{u}$.

The following data-name identifiers are defined for Riemann invariants and characteristic variables:

EXPRESS specification:

```
*)
TYPE Riemann_1D_data_name = ENUMERATION OF
     (Riemann_invariant_plus,
      Riemann_invariant_minus,
      characteristic_entropy,
      characteristic_vorticity1,
      characteristic_vorticity2,
      characteristic_acoustic_plus,
```

Table G.3 – Riemann 1-D data name identifiers

Data name identifier	Description	Units
Riemann_invariant_plus	$u + 2c/(\gamma - 1)$	**L**/**T**
Riemann_invariant_minus	$u - 2c/(\gamma - 1)$	**L**/**T**
characteristic_entropy	$p' - \rho'/\bar{c}^2$	**M**/(**LT**^{2})
characteristic_vorticity1	v'	**L**/**T**
characteristic_vorticity2	w'	**L**/**T**
characteristic_acoustic_plus	$p' + u'/(\bar{\rho}\bar{c})$	**M**/(**LT**^{2})
characteristic_acoustic_minus	$p' - u'/(\bar{\rho}\bar{c})$	**M**/(**LT**^{2})

```
      characteristic_acoustic_minus);
END_TYPE;
(*
```

The required identifiers and their meanings are given in Table G.3.

G.3.3.11 bc_type

Boundary-condition types identify the equations that should be enforced at a given boundary location. The boundary-condition types are described by **bc_type**. Some members of **bc_type** completely identify the equations to impose, while others identify a general description of the class of boundary-condition equations to impose.

bc_type is a superset of at least two kinds of boundary-condition types.

The subdivision of **bc_type** is based on function. For simple boundary-conditions, the equations and data are fixed; whereas, for compound boundary-conditions different sets of equations are imposed depending on local conditions at the boundary.

bc_type is a superset of **bc_type_simple** and **bc_type_compound**. It identifies the boundary-condition (simple or compound) at a boundary location.

EXPRESS specification:

```
*)
TYPE bc_type = SELECT
     (bc_type_simple,
      bc_type_compound);
END_TYPE;
(*
```

G.3.3.12 bc_type_simple

bc_type_simple is an enumeration type that identifies the simple boundary-condition at a boundary location.

In the descriptions below, Q is the solution vector, $\vec{q}$ is the velocity vector whose magnitude is q, the unit normal to the boundary is $\hat{n}$, and $\partial()/\partial n = \hat{n} \cdot \nabla$ is differentiation normal to the boundary.

For inflow/outflow boundary-condition descriptions, 3-D inviscid compressible flow is assumed; the 2-D equivalent should be obvious. These same boundary-conditions are typically used for viscous cases also. This '3-D Euler' assumption will be noted wherever used.

EXPRESS specification:

```
*)
TYPE bc_type_simple = EXTENSIBLE ENUMERATION OF
     (unspecified,
      application_defined,
      bc_general,
      bc_Dirichlet,
      bc_Neumann,
      bc_extrapolate,
      bc_symmetry_plane,
      bc_wall_inviscid,
      bc_wall_viscous_heat_flux,
      bc_wall_viscous_isothermal,
      bc_wall_viscous,
      bc_wall,
      bc_inflow_subsonic,
      bc_inflow_supersonic,
      bc_outflow_subsonic,
      bc_outflow_supersonic,
      bc_tunnel_inflow,
      bc_tunnel_outflow,
      bc_degenerate_line,
      bc_degenerate_point,
      bc_symmetry_polar,
      bc_axissymmetric_wedge);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: condition is unspecified;

application_defined: condition is specified via an external agreement between the data creator and the data user;

bc_general: arbitrary conditions on Q or $\partial Q/\partial n$;

bc_Dirichlet: Dirichlet condition on Q vector;

bc_Neumann: Neumann condition on $\partial Q/\partial n$;

bc_extrapolate: extrapolate Q from interior;

bc_symmetry_plane: symmetry plane; face should be coplanar

— density, pressure: $\partial()/\partial n = 0$

— tangential velocity: $\partial(\vec{q} \times \hat{n})/\partial n = 0$

— normal velocity: $\vec{q} \cdot \hat{n} = 0$

bc_wall_inviscid: inviscid (slip) wall

— normal velocity specified (default: $\vec{q} \cdot \hat{n} = 0$)

bc_wall_viscous_heat_flux: viscous no-slip wall with heat flux

— velocity Dirichlet (default: $q = 0$)

— temperature Neumann (default: adiabatic, $\partial T/\partial n = 0$)

bc_wall_viscous_isothermal: viscous no-slip, isothermal wall

— velocity Dirichlet (default: $q = 0$)

— temperature Dirichlet

bc_wall_viscous: viscous no-slip wall; special cases are **bc_wall_viscous_heat_flux** and **bc_wall_viscous_isothermal**.

— velocity Dirichlet (default: $q = 0$)

— Dirichlet or Neumann on temperature

bc_wall: general wall condition; special cases are **bc_wall_inviscid**, **bc_wall_viscous**, **bc_wall_viscous_heat_flux**, and **bc_wall_viscous_isothermal**

bc_inflow_subsonic: inflow with subsonic normal velocity

— specify 4; extrapolate 1 (3-D Euler)

bc_inflow_supersonic: inflow with supersonic normal velocity

— specify 5; extrapolate 0 (3-D Euler)

same as **bc_Dirichlet**

bc_outflow_subsonic: outflow with subsonic normal velocity

— specify 1; extrapolate 0 (3-D Euler)

bc_outflow_supersonic: outflow with supersonic normal velocity

— specify 0; extrapolate 5 (3-D Euler)

same as **bc_Extrapolate**

bc_tunnel_inflow: tunnel inlet (subsonic normal velocity)

— specify cross-flow velocity, stagnation enthalpy, entropy

— extrapolate 1 (3-D Euler)

bc_tunnel_outflow: tunnel exit (subsonic normal velocity)

— specify static pressure

— extrapolate 4 (3-D Euler)

bc_degenerate_line: face degenerated to a line;

bc_degenerate_point: face degenerated to a point;

bc_symmetry_polar: polar-coordinate singularity line; special case of **bc_degenerate_line** where degenerate face is a straight line and flowfield has polar symmetry; $\hat{s}$ is singularity line tangential unit vector

— normal velocity: $\vec{q} \times \hat{s} = 0$

— all others: $\partial()/\partial n = 0$, n normal to $\hat{s}$

bc_axissymmetric_wedge: axisymmetric wedge; special case of **bc_degenerate_line** where degenerate face is a straight line

G.3.3.13 bc_type_compound

bc_type_compound is an enumeration type that identifies the compound boundary-condition at a boundary location.

EXPRESS specification:

```
*)
TYPE bc_type_compound = ENUMERATION OF
     (unspecified,
      application_defined,
      bc_inflow,
      bc_outflow,
      bc_farfield);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: condition is unspecified;

application_defined: condition is specified via an external agreement between the data creator and the data user;

bc_inflow: inflow, arbitrary normal Mach
test on normal Mach, then perform one of: **bc_inflow_subsonic**, **bc_inflow_supersonic**;

bc_outflow: outflow, arbitrary normal Mach
test on normal Mach, then perform one of: **bc_outflow_subsonic**, **bc_outflow_supersonic**;

bc_farfield: farfield inflow/outflow, arbitrary normal Mach
test on normal velocity and normal Mach, then perform one of: **bc_inflow_subsonic**, **bc_inflow_supersonic**, **bc_outflow_subsonic**, **bc_outflow_supersonic**.

G.3.3.14 ijk_minmax

An enumeration of i-min through k-max.

NOTE See Table G.8 for one use of this type.

EXPRESS specification:

```
*)
TYPE ijk_minmax = ENUMERATION OF
     (i_min,
      j_min,
      k_min,
      i_max,
      j_max,
      k_max);
END_TYPE;
(*
```

G.3.3.15 force_moment_data_name

force_moment_data_name is an enumeration of standardized force and moment data.

Conventions for data-name identifiers for forces and moments are defined in this subclause.

Given a differential force $\vec{f}$ (i.e. a force per unit area), the force integrated over a surface is,

$$\vec{F} = F_x\hat{e}_x + F_y\hat{e}_y + F_z\hat{e}_z = \int \vec{f}\, dA,$$

where $\hat{e}_x$, $\hat{e}_y$ and $\hat{e}_z$ are the unit vectors in the x, y and z directions, respectively. The moment about a point $\vec{r}_0$ integrated over a surface is,

$$\vec{M} = M_x\hat{e}_x + M_y\hat{e}_y + M_z\hat{e}_z = \int (\vec{r} - \vec{r}_0) \times \vec{f}\, dA.$$

Lift and drag components of the integrated force are,

$$L = \vec{F} \cdot \hat{L} \qquad D = \vec{F} \cdot \hat{D}$$

where $\hat{L}$ and $\hat{D}$ are the unit vectors in the positive lift and drag directions, respectively.

Lift, drag and moment are often computed in auxiliary coordinate frames (e.g. wind axes or stability axes). We introduce the convention that lift, drag and moment are computed in the (ξ, η, ζ) coordinate system. Positive drag is assumed parallel to the ξ-direction (i.e. $\hat{D} = \hat{e}_\xi$); and positive lift is assumed parallel to the η-direction (i.e. $\hat{L} = \hat{e}_\eta$). Thus, forces and moments defined in this auxiliary coordinate system are,

$$L = \vec{F} \cdot \hat{e}_\eta \qquad D = \vec{F} \cdot \hat{e}_\xi$$

$$\vec{M} = M_\xi\hat{e}_\xi + M_\eta\hat{e}_\eta + M_\zeta\hat{e}_\zeta = \int (\vec{r} - \vec{r}_0) \times \vec{f}\, dA.$$

Lift, drag and moment coefficients in 3–D are defined as,

$$C_L = \frac{L}{\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2 S} \qquad C_D = \frac{D}{\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2 S} \qquad \vec{C}_M = \frac{\vec{M}}{\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2 c_{\mathrm{ref}} S_{\mathrm{ref}}},$$

where $\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2$ is a reference dynamic pressure, S_{ref} is a reference area, and c_{ref} is a reference length. For a wing, S_{ref} is typically the wing area and c_{ref} is the mean aerodynamic chord. In 2–D, the sectional force coefficients are,

$$c_l = \frac{L'}{\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2 c_{\mathrm{ref}}} \qquad c_d = \frac{D'}{\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2 c_{\mathrm{ref}}} \qquad \vec{c}_m = \frac{\vec{M}'}{\frac{1}{2}\rho_{\mathrm{ref}} q_{\mathrm{ref}}^2 c_{\mathrm{ref}}^2},$$

where the forces are integrated along a contour (e.g. an airfoil cross-section) rather than a surface.

The following data-name identifiers and definitions are provided for forces and moments and their associated coefficients. For coefficients, the dynamic pressure and length scales used in the normalization are provided.

Table G.4 – Force and moment data name identifiers

Data name identifier	Description	Units
force_x	$F_x = \vec{F} \cdot \hat{e}_x$	$\mathbf{ML/T}^2$
force_y	$F_y = \vec{F} \cdot \hat{e}_y$	$\mathbf{ML/T}^2$
force_z	$F_z = \vec{F} \cdot \hat{e}_z$	$\mathbf{ML/T}^2$
force_r	$F_r = \vec{F} \cdot \hat{e}_r$	$\mathbf{ML/T}^2$
force_theta	$F_\theta = \vec{F} \cdot \hat{e}_\theta$	$\mathbf{ML/T}^2$
force_phi	$F_\phi = \vec{F} \cdot \hat{e}_\phi$	$\mathbf{ML/T}^2$
moment_x	$M_x = \vec{M} \cdot \hat{e}_x$	$\mathbf{ML}^2/\mathbf{T}^2$
moment_y	$M_y = \vec{M} \cdot \hat{e}_y$	$\mathbf{ML}^2/\mathbf{T}^2$
moment_z	$M_z = \vec{M} \cdot \hat{e}_z$	$\mathbf{ML}^2/\mathbf{T}^2$
moment_r	$M_r = \vec{M} \cdot \hat{e}_r$	$\mathbf{ML}^2/\mathbf{T}^2$
moment_theta	$M_\theta = \vec{M} \cdot \hat{e}_\theta$	$\mathbf{ML}^2/\mathbf{T}^2$
moment_phi	$M_\phi = \vec{M} \cdot \hat{e}_\phi$	$\mathbf{ML}^2/\mathbf{T}^2$
moment_xi	$M_\xi = \vec{M} \cdot \hat{e}_\xi$	$\mathbf{ML}^2/\mathbf{T}^2$
moment_eta	$M_\eta = \vec{M} \cdot \hat{e}_\eta$	$\mathbf{ML}^2/\mathbf{T}^2$
moment_zeta	$M_\zeta = \vec{M} \cdot \hat{e}_\zeta$	$\mathbf{ML}^2/\mathbf{T}^2$
moment_center_x	$x_0 = \vec{r}_0 \cdot \hat{e}_x$	**L**
moment_center_y	$y_0 = \vec{r}_0 \cdot \hat{e}_y$	**L**
moment_center_z	$z_0 = \vec{r}_0 \cdot \hat{e}_z$	**L**
lift	L or L'	$\mathbf{ML/T}^2$
drag	D or D'	$\mathbf{ML/T}^2$
coef_lift	C_L or c_l	-
coef_drag	C_D or c_d	-
coef_moment_x	$\vec{C}_M \cdot \hat{e}_x$ or $\vec{c}_m \cdot \hat{e}_x$	-
coef_moment_y	$\vec{C}_M \cdot \hat{e}_y$ or $\vec{c}_m \cdot \hat{e}_y$	-
coef_moment_z	$\vec{C}_M \cdot \hat{e}_z$ or $\vec{c}_m \cdot \hat{e}_z$	-
coef_moment_r	$\vec{C}_M \cdot \hat{e}_r$ or $\vec{c}_m \cdot \hat{e}_r$	-
coef_moment_theta	$\vec{C}_M \cdot \hat{e}_\theta$ or $\vec{c}_m \cdot \hat{e}_\theta$	-
coef_moment_phi	$\vec{C}_M \cdot \hat{e}_\phi$ or $\vec{c}_m \cdot \hat{e}_\phi$	-
coef_moment_xi	$\vec{C}_M \cdot \hat{e}_\xi$ or $\vec{c}_m \cdot \hat{e}_\xi$	-
coef_moment_eta	$\vec{C}_M \cdot \hat{e}_\eta$ or $\vec{c}_m \cdot \hat{e}_\eta$	-
coef_moment_zeta	$\vec{C}_M \cdot \hat{e}_\zeta$ or $\vec{c}_m \cdot \hat{e}_\zeta$	-
coef_pressure_dynamic	$1/2\rho_{\text{ref}} q_{\text{ref}}^2$	$\mathbf{M/(LT}^2)$
coef_area	S_{ref}	$\mathbf{L}^2$
coef_length	c_{ref}	**L**

EXPRESS specification:

```
*)
TYPE force_moment_data_name = ENUMERATION OF
     (force_x,
      force_y,
      force_z,
      force_r,
      force_theta,
      force_phi,
      moment_x,
      moment_y,
      moment_z,
      moment_r,
      moment_theta,
      moment_phi,
      moment_xi,
      moment_eta,
      moment_zeta,
      moment_center_x,
      moment_center_y,
      moment_center_z,
      lift,
      drag,
      coef_lift,
      coef_drag,
      coef_moment_x,
      coef_moment_y,
      coef_moment_z,
      coef_moment_r,
      coef_moment_theta,
      coef_moment_phi,
      coef_moment_xi,
      coef_moment_eta,
      coef_moment_zeta,
      coef_moment_pressure_dynamic,
      coef_moment_area,
      coef_length);
END_TYPE;
(*
```

The meanings of the enumerated items are given in Table G.4.

G.3.3.16 governing_equation_type

governing_equation_type is an enumeration of the classes of equations.

EXPRESS specification:

```
*)
TYPE governing_equation_type = ENUMERATION OF
     (unspecified,
      application_defined,
      full_potential,
      Euler,
      NS_laminar,
      NS_turbulent,
      NS_laminar_incompressible,
      NS_turbulent_incompressible);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: is unspecified;

application_defined: is specified via an external agreement between the data creator and the data user;

full_potential: is full potential flow;

Euler: is Euler flow;

NS_laminar: is Navier-Stokes laminar flow;

NS_turbulent: is Navier-Stokes turbulent flow;

NS_laminar_incompressible: is Navier-Stokes laminar incrompressible flow;

NS_turbulent_incompressible: is Navier-Stokes turbulent incrompressible flow;

G.3.3.17 gas_model_type

gas_model_type is an enumeration of the gaseous state models relating pressure, temperature and density.

EXPRESS specification:

```
*)
TYPE gas_model_type = ENUMERATION OF
     (unspecified,
      application_defined,
      ideal,
      Van_der_Waals);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: is unspecified.

application_defined: is specified via an external agreement between the data creator and the data user;

ideal: the state model is the perfect gas law. The pressure, temperature and density are related by,

$$p = \rho RT,$$

where R is the ideal gas constant. Related quantities are the specific heat at constant pressure (c_p), specific heat at constant volume (c_v) and specific heat ratio ($\gamma = c_p/c_v$). The gas constant and specific heats are related by $R = c_p - c_v$.

The **standard_data_name** identifiers associated with the perfect gas law are: **ideal_gas_constant**, **specific_heat_ratio**, **specific_heat_volume** and **specific_heat_pressure**. These are described in clause G.3.3.18.

Van_der_Waals: the state model is Van der Waals' equation.

G.3.3.18 gas_model_data_name

gas_model_data_name is an enumeration of the standard data names that may be associated with data corresponding to different kinds of gaseous state models.

EXPRESS specification:

```
*)
TYPE gas_model_data_name = ENUMERATION OF
     (ideal_gas_constant,
      specific_heat_ratio,
      specific_heat_pressure,
      specific_heat_volume);
END_TYPE;
(*
```

Enumerated item definitions:

ideal_gas_constant: indicates the ideal gas constant;

specific_heat_ratio: indicates the ratio of the specific heats at constant pressure to constant volume;

specific_heat_pressure: indicates the specific heat at constant pressure;

specific_heat_volume: indicates the specific heat at constant volume.

G.3.3.19 viscosity_model_type

viscosity_model_type is an enumeration of the relationships between molecular viscosity and temperature.

EXPRESS specification:

```
*)
TYPE viscosity_model_type = ENUMERATION OF
     (unspecified,
      application_defined,
      constant_viscosity,
      power_law,
      Sutherland_law);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: is unspecified;

application_defined: is specified via an external agreement between the data creator and the data user;

constant_viscosity: the molecular viscosity is constant throughout the field and is equal to some reference value ($\mu = \mu_{\text{ref}}$).

The standard data name identifier associated with this model is **viscosity_molecular_reference** and is described in clause G.3.3.20.

power_law: the molecular viscosity follows a power-law relation,

$$\mu = \mu_{\text{ref}} \left(\frac{T}{T_{\text{ref}}} \right)^n .$$

The standard data name identifiers associated with this model are: **power_law_exponent**, **temperature_reference** and **viscosity_molecular_reference**. These are described in clause G.3.3.20 and clause G.3.3.22.

Sutherland_law: Sutherland's Law for molecular viscosity,

$$\mu = \mu_{\text{ref}} \left(\frac{T}{T_{\text{ref}}} \right)^{3/2} \frac{T_{\text{ref}} + T_s}{T + T_s},$$

where T_s is the Sutherland Law constant, and μ_{ref} and T_{ref} are the reference viscosity and temperature, respectively.

The standard data name identifiers associated with this model are: **Sutherland_constant_viscosity**, **temperature_reference** and **viscosity_molecular_reference**. These are described in clause G.3.3.20 and clause G.3.3.22.

NOTE 1

For air [4], the power-law exponent is $n = 0.666$, Sutherlands Law constant (T_s) is 110.6 K, the reference temperature (T_{ref}) is 273.15 K, and the reference viscosity (μ_{ref}) is 1.716×10^{-5} kg/(m s).

G.3.3.20 viscosity_model_data_name

viscosity_model_data_name is an enumeration of the standard data names that may be associated with data corresponding to different kinds of viscosity models.

EXPRESS specification:

```
*)
TYPE viscosity_model_data_name = ENUMERATION OF
     (viscosity_molecular_reference,
      Sutherland_constant_viscosity);
END_TYPE;
(*
```

Enumerated item definitions:

viscosity_molecular_reference: indicates a molecular viscosity reference;

Sutherland_constant_viscosity: indicates Sutherland's constant for Sutherland's Law for molecular viscosity,

G.3.3.21 thermal_conductivity_model_type

thermal_conductivity_model_type is an enumeration of the relationships between the thermal-conductivity coefficient and temperature.

EXPRESS specification:

```
*)
TYPE thermal_conductivity_model_type = ENUMERATION OF
     (unspecified,
      application_defined,
      independent,
      power_law,
      Sutherland_law,
      constant_Prandtl);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: is unspecified;

application_defined: is specified via an external agreement between the data creator and the data user;

independent: identifies that the thermal conductivity coefficient is independent of the temperature and is equal to some reference value ($k = k_{\text{ref}}$).

The standard data name identifier associated with this model is **thermal_conductivity_reference**. This is described in clause G.3.3.22.

power_law: the thermal conductivity is related to temperature via a power-law.

$$k = k_{\text{ref}} \left(\frac{T}{T_{\text{ref}}} \right)^n .$$

The standard data name identifiers associated with this model are: **power_law_exponent**, **temperature_reference** and **thermal_conductivity_reference**. These are described in clause G.3.3.22.

Sutherland_law: Sutherland's Law for thermal conductivity.

$$k = k_{\text{ref}} \left(\frac{T}{T_{\text{ref}}} \right)^{3/2} \frac{T_{\text{ref}} + T_s}{T + T_s},$$

where T_s is the Sutherland Law constant, and k_{ref} and T_{ref} are the reference thermal conductivity and temperature, respectively.

The standard data name identifiers associated with this model are: **Sutherland_constant_conductivity**, **temperature_reference** and **thermal_conductivity_reference**. These are described in clause G.3.3.22.

Table G.5 – Thermal conductivity model data name identifiers

Data name identifier	Description	Units
power_law_exponent	n	-
temperature_reference	T_{ref}	Θ
thermal_conductivity_reference	k_{ref}	$\mathbf{ML/(T^2}\Theta)$
Sutherland_constant_conductivity	T_s	Θ
constant_Prandtl	P_r	-

constant_Prandtl: the Prandtl number ($Pr = \mu c_p/k$) is constant and equal to some reference value.

The standard data name identifier associated with this model is **constant_Prandtl**, and is described in clause G.3.3.22.

NOTE

For air [4], the Prandtl number is $Pr = 0.72$, the power-law exponent is $n = 0.81$, Sutherlands Law constant (T_s) is 194.4 K, the reference temperature (T_{ref}) is 273.15 K, and the reference thermal conductivity (k_{ref}) is 2.414×10^{-2} $kg\ m/(s^3K)$.

G.3.3.22 thermal_conductivity_model_data_name

thermal_conductivity_model_data_name is an enumeration of the standard data names that may be associated with data corresponding to different kinds of thermal conductivity models. The identifiers and their units are given in Table G.5.

EXPRESS specification:

```
*)
TYPE thermal_conductivity_model_data_name = ENUMERATION OF
     (power_law_exponent,
      temperature_reference,
      thermal_conductivity_reference,
      Sutherland_constant_conductivity,
      constant_Prandtl);
END_TYPE;
(*
```

Enumerated item definitions:

power_law_exponent: indicates a power law exponent;

temperature_reference: indicates a reference temperature;

thermal_conductivity_reference: indicates a reference thermal conductivity;

Sutherland_constant_conductivity: indicates Sutherland's Law constant for thermal conductivity;

thermal_conductivity_molecular_reference: indicates a reference molecular thermal conductivity;

constant_Prandtl: indicates a Prandtl number.

G.3.3.23 turbulence_closure_type

turbulence_closure_type is an enumeration of the kinds of turbulence closure for the Reynolds stress terms of the Navier-Stokes equations.

EXPRESS specification:

```
*)
TYPE turbulence_closure_type = ENUMERATION OF
     (unspecified,
      application_defined,
      eddy_viscosity,
      Reynolds_stress,
      Reynolds_stress_algebraic);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: is unspecified;

application_defined: is specified via an external agreement between the data creator and the data user;

eddy_viscosity: Boussinesq eddy-velocity closure. The Reynolds stresses are approximated as the product of an eddy viscosity (ν_t) and the mean strain tensor. Using indicial notation, the relation is,

$$-\overline{u_i\prime u_j\prime} = \nu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

where $-\overline{u_i\prime u_j\prime}$ are the Reynolds stresses.

Reynolds_stress: no approximation of the Reynolds stresses.

Reynolds_stress_algebraic: an algebraic approximation for the Reynolds stresses based on some intermediate transport quantities.

The associated **standard_data_name** name identifiers are: **eddy_viscosity** and **Prandtl_turbulent**. These are described in clause G.3.3.24.

G.3.3.24 turbulence_closure_data_name

turbulence_closure_data_name is an enumeration of the standard data names that may be associated with data corresponding to different kinds of turbulence enclosures.

EXPRESS specification:

```
*)
TYPE turbulence_closure_data_name = ENUMERATION OF
     (eddy_viscosity,
      Prandtl_turbulent);
END_TYPE;
(*
```

Enumerated item definitions:

eddy_viscosity: indicates an eddy viscosity;

Prandtl_turbulent: indicates a Prandtl turbulent number.

G.3.3.25 turbulence_model_type

turbulence_model_type is an enumeration of the equation sets for modeling the turbulence quantities.

EXPRESS specification:

```
*)
TYPE turbulence_model_type = ENUMERATION OF
     (unspecified,
      application_defined,
      algebraic_Baldwin_Lomax,
      algebraic_Cebeci_Smith,
      half_equation_Johnson_King,
      one_equation_Baldwin_Barth,
      one_equation_Spalart_Allmaras,
      two_equation_Jones_Launder,
      two_equation_Menter_SST,
      two_equation_Wilcox);
END_TYPE;
(*
```

Enumerated item definitions:

unspecified: is unspecified;

application_defined: is specified via an external agreement between the data creator and the data user;

algebraic_Baldwin_Lomax: is Baldwin-Lomax;

algebraic_Cebeci_Smith: is Cebeci-Smith;

half_equation_Johnson_King: is Johnson-King;

one_equation_Baldwin_Barth: is Baldwin-Barth;

one_equation_Spalart_Allmaras: is Spalart-Allmaras;

two_equation_Jones_Launder: is Jones-Launder;

two_equation_Menter_SST: is Menter;

two_equation_Wilcox: is Wilcox.

The associated **standard_data_name** name identifiers for the Spalart-Allmaras turbulence model (version Ia) are: **turbulent_SA_cb1**, **turbulent_SA_cb2**, **turbulent_SA_sigma**, **turbulent_SA_kappa**, **turbulent_SA_cw1**, **turbulent_SA_cw2**, **turbulent_SA_cw3**, **turbulent_SA_cv1**, **turbulent_SA_ct1**, **turbulent_SA_ct2**, **turbulent_SA_ct3**, and **turbulent_SA_ct4**. These are described in clause G.3.3.26.

Table G.6 – Turbulence data name identifiers

Data name identifier	Description	Units
turbulent_distance	distance to nearest wall	**L**
turbulent_energy_kinetic	$k = \frac{1}{2}(\overline{u'u'} + \overline{v'v'} + \overline{w'w'})$	$\mathbf{L}^2/\mathbf{T}^2$
turbulent_dissipation	ϵ	$\mathbf{L}^2/\mathbf{T}^3$
turbulent_dissipation_rate	ϵ/k	$\mathbf{T}^{-1}$
turbulent_BB_Reynolds	Baldwin-Barth one-equation model R_T	-
turbulent_SA_nu_tilde	Spalart-Allmaras one-equation model $\tilde{\nu}$	$\mathbf{L}^2/\mathbf{T}$
turbulent_SA_chi	S-A model $\chi = \tilde{\nu}/\nu$	-
turbulent_SA_cb1	S-A model $c_{b1} = 0.1355$	-
turbulent_SA_cb2	S-A model $c_{b2} = 0.622$	-
turbulent_SA_sigma	S-A model $\sigma = 2/3$	-
turbulent_SA_kappa	S-A model $\kappa = 0.41$ (von Karman constant)	-
turbulent_SA_cw1	S-A model $c_{w1} = 3.2391$	-
turbulent_SA_cw2	S-A model $c_{w2} = 0.3$	-
turbulent_SA_cw3	S-A model $c_{w3} = 2$	-
turbulent_SA_cv1	S-A model $c_{v1} = 7.1$	-
turbulent_SA_ct1	S-A model $c_{t1} = 1$	-
turbulent_SA_ct2	S-A model $c_{t2} = 2$	-
turbulent_SA_ct3	S-A model $c_{t3} = 1.2$	-
turbulent_SA_ct4	S-A model $c_{t4} = 0.5$	-

G.3.3.26 turbulence_model_data_name

turbulence_model_data_name is an enumeration of standardized Reynolds-averaged Navier-Stokes turbulence model variables.

Turbulence model solution quantities and model constants present a particularly difficult nomenclature problem — to be precise it is necessary to identify both the variable and the model (and version) that it comes from. The list (in Table G.6) falls short in this respect.

EXPRESS specification:

```
*)
TYPE turbulence_model_data_name = ENUMERATION OF
     (turbulent_distance,
      turbulent_energy_kinetic,
      turbulent_dissipation,
      turbulent_dissipation_rate,
      turbulent_BB_Reynolds,
      turbulent_SA_nu_tilde,
      turbulent_SA_chi,
      turbulent_SA_cb1,
      turbulent_SA_cb2,
      turbulent_SA_sigma,
      turbulent_SA_kappa,
      turbulent_SA_cw1,
      turbulent_SA_cw2,
      turbulent_SA_cw3,
      turbulent_SA_cv1,
      turbulent_SA_ct1,
      turbulent_SA_ct2,
      turbulent_SA_ct3,
      turbulent_SA_ct4);
END_TYPE;
(*
```

The required identifiers and their meanings are given in Table G.6.

G.3.3.27 flow_solution_data_name

flow_solution_data_name is an enumeration of standardized flow solution data.

This clause describes data-name identifiers for typical Navier-Stokes solution variables. The list is obviously incomplete, but should suffice for initial implementation of the CFD system. The variables listed in this section are dimensional or raw quantities; nondimensional parameters and coefficients based on these variables are discussed in G.3.3.6.

A reasonably universal notation is used for state variables. Static quantities are measured with the fluid at speed: static density (ρ), static pressure (p), static temperature (T), static internal energy per unit mass (e), static enthalpy per unit mass (h), entropy (s), and static speed of sound (c). The true entropy isaa approximated by the function $\tilde{s} = p/\rho^\gamma$ (this assumes an ideal gas). The velocity is $\vec{q} = u\hat{e}_x + v\hat{e}_y + w\hat{e}_z$, with magnitude $q = \sqrt{\vec{q}\cdot\vec{q}}$. Stagnation quantities are obtained by bringing the fluid isentropically to rest; these are identified by a subscript '$_0$'. The term 'total' is also used to refer to stagnation quantities.

Conservation variables are density, momentum ($\rho\vec{q} = \rho u\hat{e}_x + \rho v\hat{e}_y + \rho w\hat{e}_z$), and stagnation energy per unit volume (ρe_0).

Molecular diffusion and heat transfer introduce the molecular viscosity (μ), kinematic viscosity (ν) and thermal conductivity coefficient (k). These are obtained from the state variables through auxiliary correlations. For a perfect gas, μ and k are functions of static temperature only.

The Navier-Stokes equations involve the strain tensor ($\bar{\bar{S}}$) and the shear-stress tensor ($\bar{\bar{\tau}}$). Using indicial notation, the 3–D cartesian components of the strain tensor are,

$$\bar{\bar{S}}_{i,j} = \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right),$$

and the stress tensor is,

$$\bar{\bar{\tau}}_{i,j} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \lambda\frac{\partial u_k}{\partial x_k},$$

where $(x_1, x_2, x_3) = (x, y, z)$ and $(u_1, u_2, u_3) = (u, v, w)$. The bulk viscosity is usually approximated as $\lambda = -2/3\mu$.

Reynolds averaging of the Navier-Stokes equations introduce Reynolds stresses ($-\rho\overline{u'v'}$, etc.) and turbulent heat flux terms ($-\rho\overline{u'e'}$, etc.), where primed quantities are instantaneous fluctuations and the bar is an averaging operator. These quantities are obtained from auxiliary turbulence closure models. Reynolds-stress models formulate transport equations for the Reynolds stresses directly; whereas, eddy-viscosity models correlate the Reynolds stresses with the mean strain rate,

$$-\overline{u'v'} = \nu_t\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right),$$

where ν_t is the eddy viscosity. The eddy viscosity is either correlated to mean flow quantities by algebraic models or by auxiliary transport models. An example two-equation turbulence transport model is the k-ϵ model, where transport equations are formulated for the turbulent kinetic energy ($k = \frac{1}{2}(\overline{u'u'} + \overline{v'v'} + \overline{w'w'})$) and turbulent dissipation (ϵ).

Skin friction evaluated at a surface is the dot product of the shear stress tensor with the surface normal:

$$\vec{\tau} = \bar{\bar{\tau}} \cdot \hat{n},$$

Note that skin friction is a vector.

The following data-name identifiers are defined for flow-solution quantities.

Table G.7 – Flow solution data name identifiers

Data name identifier	**Description**	**Units**
potential	potential: $\nabla\phi = \vec{q}$	$\mathbf{L}^2/\mathbf{T}$
stream_function	stream function (2–D): $\nabla\times\psi = \vec{q}$	$\mathbf{L}^2/\mathbf{T}$
density	static density (ρ)	$\mathbf{M}/\mathbf{L}^3$
pressure	static pressure (p)	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
temperature	static temperature (T)	Θ
energy_internal	static internal energy per unit mass (e)	$\mathbf{L}^2/\mathbf{T}^2$
enthalpy	static enthalpy per unit mass (h)	$\mathbf{L}^2/\mathbf{T}^2$
entropy	entropy (s)	$\mathbf{M}\mathbf{L}^2/(\mathbf{T}^2\Theta)$
entropy_approx	approximate entropy ($\tilde{s} = p/\rho^\gamma$)	$\mathbf{L}^{3\gamma-1}/(\mathbf{M}^{\gamma-1}\mathbf{T}^2)$
density_stagnation	stagnation density (ρ_0)	$\mathbf{M}/\mathbf{L}^3$
pressure_stagnation	stagnation pressure (p_0)	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
temperature_stagnation	stagnation temperature (T_0)	Θ
energy_stagnation	stagnation energy per unit mass (e_0)	$\mathbf{L}^2/\mathbf{T}^2$
enthalpy_stagnation	stagnation enthalpy per unit mass (h_0)	$\mathbf{L}^2/\mathbf{T}^2$
energy_stagnation_density	stagnation energy per unit volume (ρe_0)	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
velocity_x	x-component of velocity ($u = \vec{q}\cdot\hat{e}_x$)	$\mathbf{L}/\mathbf{T}$
velocity_y	y-component of velocity ($v = \vec{q}\cdot\hat{e}_y$)	$\mathbf{L}/\mathbf{T}$
velocity_z	z-component of velocity ($w = \vec{q}\cdot\hat{e}_z$)	$\mathbf{L}/\mathbf{T}$
velocity_r	radial velocity component ($\vec{q}\cdot\hat{e}_r$)	$\mathbf{L}/\mathbf{T}$
velocity_theta	velocity component in θ direction ($\vec{q}\cdot\hat{e}_\theta$)	$\mathbf{L}/\mathbf{T}$
velocity_phi	velocity component in ϕ direction ($\vec{q}\cdot\hat{e}_\phi$)	$\mathbf{L}/\mathbf{T}$
velocity_magnitude	velocity magnitude ($q = \sqrt{\vec{q}\cdot\vec{q}}$)	$\mathbf{L}/\mathbf{T}$
velocity_normal	normal velocity component ($\vec{q}\cdot\hat{n}$)	$\mathbf{L}/\mathbf{T}$
velocity_tangential	tangential velocity component (2–D)	$\mathbf{L}/\mathbf{T}$
velocity_sound	static speed of sound	$\mathbf{L}/\mathbf{T}$
velocity_sound_stagnation	stagnation speed of sound	$\mathbf{L}/\mathbf{T}$
momentum_x	x-component of momentum (ρu)	$\mathbf{M}/(\mathbf{L}^2\mathbf{T})$
momentum_y	y-component of momentum (ρv)	$\mathbf{M}/(\mathbf{L}^2\mathbf{T})$
momentum_z	z-component of momentum (ρw)	$\mathbf{M}/(\mathbf{L}^2\mathbf{T})$

Continued on next page

Table G.7 — concluded from previous page

Data name identifier	Description	Units
momentum_magnitude	magnitude of momentum (ρq)	$\mathbf{M}/(\mathbf{L}^2\mathbf{T})$
energy_kinetic	$\frac{1}{2}(u^2 + v^2 + w^2) = \frac{1}{2}q^2$	$\mathbf{L}^2/\mathbf{T}^2$
pressure_dynamic	$\frac{1}{2}\rho q^2$	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
vorticity_x	$\omega_x = \partial w/\partial y - \partial v/\partial z = \vec{\omega}\cdot\hat{e}_x$	$\mathbf{T}^{-1}$
vorticity_y	$\omega_y = \partial u/\partial z - \partial w/\partial x = \vec{\omega}\cdot\hat{e}_y$	$\mathbf{T}^{-1}$
vorticity_z	$\omega_z = \partial v/\partial x - \partial u/\partial y = \vec{\omega}\cdot\hat{e}_z$	$\mathbf{T}^{-1}$
vorticity_magnitude	$\omega = \sqrt{\vec{\omega}\cdot\vec{\omega}}$	$\mathbf{T}^{-1}$
skin_friction_x	x-component of skin friction ($\vec{\tau}\cdot\hat{e}_x$)	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
skin_friction_y	y-component of skin friction ($\vec{\tau}\cdot\hat{e}_y$)	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
skin_friction_z	z-component of skin friction ($\vec{\tau}\cdot\hat{e}_z$)	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
skin_friction_magnitude	skin friction magnitude ($\sqrt{\vec{\tau}\cdot\vec{\tau}}$)	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
velocity_angle_x	velocity angle ($\arccos(u/q) \in [0, 180^\circ)$)	α
velocity_angle_y	$\arccos(v/q)$	α
velocity_angle_z	$\arccos(w/q)$	α
velocity_unit_vector_x	x-component of velocity unit vector ($(\vec{q}\cdot\hat{e}_x)/q$)	-
velocity_unit_vector_y	y-component of velocity unit vector ($(\vec{q}\cdot\hat{e}_y)/q$)	-
velocity_unit_vector_z	z-component of velocity unit vector ($(\vec{q}\cdot\hat{e}_z)/q$)	-
mass_flow	mass flow normal to a plane ($\rho\vec{q}\cdot\hat{n}$)	$\mathbf{M}/(\mathbf{L}^2\mathbf{T})$
viscosity_kinematic	kinematic viscosity ($\nu = \mu/\rho$)	$\mathbf{L}^2/\mathbf{T}$
viscosity_molecular	molecular viscosity (μ)	$\mathbf{M}/(\mathbf{L}\mathbf{T})$
viscosity_eddy	eddy viscosity (ν_t)	$\mathbf{L}^2/\mathbf{T}$
thermal_conductivity	thermal conductivity coefficient (k)	$\mathbf{M}\mathbf{L}/(\mathbf{T}^3\Theta)$
ideal_gas_constant	ideal gas constant ($R = c_p - c_v$)	$\mathbf{L}/(\mathbf{T}^2\Theta)$
specific_heat_pressure	specific heat at constant pressure (c_p)	$\mathbf{L}^2/(\mathbf{T}^2\Theta)$
specific_heat_volume	specific heat at constant volume (c_v)	$\mathbf{L}^2/(\mathbf{T}^2\Theta)$
Reynolds_stress_xx	Reynolds stress $-\rho\overline{u'u'}$	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
Reynolds_stress_xy	Reynolds stress $-\rho\overline{u'v'}$	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
Reynolds_stress_xz	Reynolds stress $-\rho\overline{u'w'}$	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
Reynolds_stress_yy	Reynolds stress $-\rho\overline{v'v'}$	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
Reynolds_stress_yz	Reynolds stress $-\rho\overline{v'w'}$	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$
Reynolds_stress_zz	Reynolds stress $-\rho\overline{w'w'}$	$\mathbf{M}/(\mathbf{L}\mathbf{T}^2)$

EXPRESS specification:

```
*)
TYPE flow_solution_data_name = ENUMERATION OF
     (potential,
      stream_function,
      density,
      pressure,
      temperature,
      energy_internal,
      enthalpy,
      entropy,
      entropy_approx,
      density_stagnation,
      pressure_stagnation,
      temperature_stagnation,
      energy_stagnation,
      enthalpy_stagnation,
      energy_stagnation_density,
      velocity_x,
      velocity_y,
      velocity_z,
      velocity_r,
      velocity_theta,
      velocity_phi,
      velocity_magnitude,
      velocity_normal,
      velocity_tangential,
      velocity_sound,
      velocity_sound_stagnation,
      momentum_x,
      momentum_y,
      momentum_z,
      momentum_magnitude,
      energy_kinetic,
      pressure_dynamic,
      vorticity_x,
      vorticity_y,
      vorticity_z,
      vorticity_magnitude,
      skin_friction_x,
      skin_friction_y,
      skin_friction_z,
      skin_friction_magnitude,
      velocity_angle_x,
      velocity_angle_y,
      velocity_angle_z,
      velocity_unit_vector_x,
      velocity_unit_vector_y,
      velocity_unit_vector_z,
      mass_flow,
      viscosity_kinematic,
      viscosity_molecular,
      viscosity_eddy,
      thermal_conductivity,
      ideal_gas_constant,
      specific_heat_pressure,
      specific_heat_volume,
      Reynolds_stress_xx,
      Reynolds_stress_xy,
      Reynolds_stress_xz,
      Reynolds_stress_yy,
      Reynolds_stress_yz,
      Reynolds_stress_zz);
END_TYPE;
(*
```

The meanings of the identifiers are given in Table G.7.

G.3.4 analysis_arm entity definitions

G.3.4.1 analysis_model

The highest level structure in an analysis database is **analysis_model**. It contains the dimensionality of the grid and a list of the analysis steps performed. Globally applicable information, including a reference state, a set of equations for solution, dimensional units, and convergence history may also be attached. In addition structures for describing or annotating the database are also accomodated.

EXPRESS specification:

```
*)
ENTITY analysis_model;
  annotation         : info;
  model_for          : analysis;
  behaviour_type     : OPTIONAL equation_set;
  actions            : LIST OF analysis_step;
  cell_dimension     : INTEGER;
  physical_dimension : INTEGER;
  refstate           : OPTIONAL reference_state;
  class              : data_class;
  units              : OPTIONAL dimensional_units;
  history            : OPTIONAL convergence_history;
  data               : LIST OF integral_data;
  families           : LIST OF family;
END_ENTITY;
(*
```

Attribute definitions:

annotation: is annotation;

model_for: is the **analysis** for which this is a model;

behaviour_type: is the description of the governing equations associated with the entire database. This structure contains information on the general class of governing equations, equations of state, and constants associated with the equations.

actions: is the sequence of analysis steps performed.

cell_dimension: is the dimension of cells in the mesh.

physical_dimension: is the number of coordinates required to define a node position.

refstate: is reference data applicable to the entire database;

EXAMPLE For a CFD analysis of an external flow problem, quantities such as Reynolds number and freestream Mach number are given here.

class: is the global default data class for the database. If the analysis database contains dimensional data (e.g., velocity with units of m/s), **units** shall be used to describe the system of units employed.

units: is the specification of the global default units;

history: is globally relevant convergence history. The convergence information includes total configuration forces, global parameters, and global residual and solution-change norms taken over all the zones.

data: is miscellaneous data. Candidates for inclusion are global forces and moments.

families: is global family information;

class, **units**, **refstate** and **behaviour_type** have a special function in the hierarchy. They are globally applicable throughout the database, but their values may be superseded by local entities (e.g., within a given step).

G.3.4.2 analysis_step

An **analysis_step** is a single step, or one of a series of steps, in an analysis. It contains a list of the zones making up the domain for the step.

EXPRESS specification:

```
*)
ENTITY analysis_step;
--  SUBTYPE OF (analysis_action);
  annotation         : info;
  cell_dimension     : INTEGER;
  physical_dimension : INTEGER;
  zones              : LIST OF zone;
  refstate           : OPTIONAL reference_state;
  class              : data_class;
  units              : OPTIONAL dimensional_units;
  equations          : OPTIONAL equation_set;
  history            : OPTIONAL convergence_history;
  data               : LIST OF integral_data;
  families           : LIST OF family;
END_ENTITY;
(*
```

Attribute definitions:

annotation: is annotation;

cell_dimension: is the dimension of cells in the mesh.

physical_dimension: is the number of coordinates required to define a node position.

zones: is data specific to each zone or block in a multiblock case. The size of the list defines the number of zones or blocks in the domain.

refstate: is reference data applicable to the entire step;

class: is the global default data class for the step. If the analysis database contains dimensional data (e.g., velocity with units of m/s), **units** shall be used to describe the system of units employed.

units: is the specification of the default units for the step;

equations: is the description of the governing equations associated with the step. This structure contains information on the general class of governing equations, equations of state, and constants associated with the equations.

history: is globally relevant convergence history. The convergence information includes total configuration forces, global parameters, and global residual and solution-change norms taken over all the zones.

data: is miscellaneous data. Candidates for inclusion are global forces and moments.

families: is global family information;

G.3.4.3 zone

An **zone** contains all information pertinent to an individual multiblock zone. This information includes the number of cells and vertices making up the grid, the physical coordinates of the grid vertices, the solution, multiblock interface connectivity, boundary-conditions, and zonal convergence-history data. In addition this structure contains a reference state, a set of governing equations, and dimensional units that are all unique to the zone.

EXPRESS specification:

```
*)
ENTITY zone;
  grid               : mesh;
  pyhsical_dimension : INTEGER;
  coordinates        : OPTIONAL grid_coordinates;
  family             : OPTIONAL family;
  solution           : LIST OF solution;
  field_data         : LIST OF discrete_data;
  global_data        : LIST OF integral_data;
  grid_connectivity  : OPTIONAL multiple_mesh_block;
  conditions         : OPTIONAL zone_bc;
  rstate             : OPTIONAL reference_state;
  dclass             : OPTIONAL data_class;
  dimunits           : OPTIONAL dimensional_units;
  equations          : OPTIONAL equation_set;
  history            : OPTIONAL convergence_history;
  nindices           : INTEGER;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_zone FOR zone;
  ABSTRACT SUPERTYPE;
  ONEOF(structured_zone,
        unstructured_zone);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

grid: is the mesh describing the topological shape of the **zone**;

physical_dimension: is the number of coordinates required to define a node position;

coordinates: are the physical coordinates of the grid vertices. This structure defines the physical shape of the **grid** (i.e., the **zone**); it may optionally contain physical coordinates of rind or ghost points.

family: a family may be used to specify geometry of certain boundary-conditions associated with the **zone**;

solution: is the solution quantities. Each instance of **solution** shall only contain data at a single grid location (vertices, cell-centers, etc.); therefore, multiple **solution** structures are provided to store solution data at different grid locations. These structures may optionally contain solution data defined at rind points.

field_data: is miscellaneous field data. Candidate information includes residuals, fluxes and other discrete data that is considered auxiliary to the solution.

global_data: is miscellaneous zone-specific global data, other than reference-state data and convergence history information.

grid_connectivity: is the multiblock interface-connectivity information.

conditions: is the boundary-condition information.

rstate: is non-default reference-state data.

dclass: is the non-default class of data.

dimunits: is the non-default system of units.

equations: is the non-default set of equations for the **zone**;

history: is the non-default convergence history of the zone; this includes residual and solution-change norms.

nindices: The number of indices required to identify uniquely a vertex or a cell in the grid. It is the indexical dimensionality of the computational grid. For structured-grid calculations, **nindices** is usually the same as the spatial problem being solved (e.g., **nindices**=3 for a 3-D problem). For lower-dimensional flowfields, such as quasi 3-D flow, **nindices** may not be the same as the dimensionality of the position vector or the velocity vector. For unstructured grids, usually **nindices**=1 since all the grid points and flow solution variables are stored in 1-D arrays. However, there are instances, such as prismatic boundary-layer grids, where **nindices** may be 2.

G.3.4.4 structured_zone

An **structured_zone** contains the information pertinent to an individual structured multiblock zone.

EXPRESS specification:

```
*)
ENTITY structured_zone
  SUBTYPE OF (zone);
  SELF\zone.grid : structured_mesh;
END_ENTITY;
(*
```

Attribute definitions:

grid: the **structured_mesh** describing the topology of the **zone**.

G.3.4.5 unstructured_zone

An **unstructured_zone** contains the information pertinent to an individual unstructured zone.

EXPRESS specification:

```
*)
```

```
ENTITY unstructured_zone
  SUBTYPE OF (zone);
  SELF\zone.grid : unstructured_mesh;
END_ENTITY;
(*
```

Attribute definitions:

grid: the **unstructured_mesh** describing the topology of the **zone**.

G.3.4.6 mesh

The **mesh** is the basis of all mesh topology representations. It consists of one or more cells and there are several ways of representing a mesh.

EXPRESS specification:

```
*)
ENTITY mesh;
--  SUBTYPE OF (topological_representation_item);
  annotation  : info;
  index_count : INTEGER;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_mesh FOR mesh;
  ABSTRACT SUPERTYPE;
  ONEOF(structured_mesh,
        unstructured_mesh);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

annotation: is annotation;

index_count: The number of indices required to identify uniquely a vertex or cell in the mesh.

G.3.4.7 structured_mesh

A **structured_mesh** has a regular topology. A **structured_mesh** has a parametric coordinate system; the parametric coordinate system for a three-dimensional structured mesh is shown in Figure G.51.

For each cell within a **structured_mesh**, the parametric coordinate system for that cell is identical to the parametric coordinate system for the mesh, except for an origin shift. The parametric coordinates of vertex (i, j, k) in a 3-D mesh with n, m and p cells in the 3 dimensions are:

$$((i-1)/n, (j-1)/m, (k-1)/p)$$

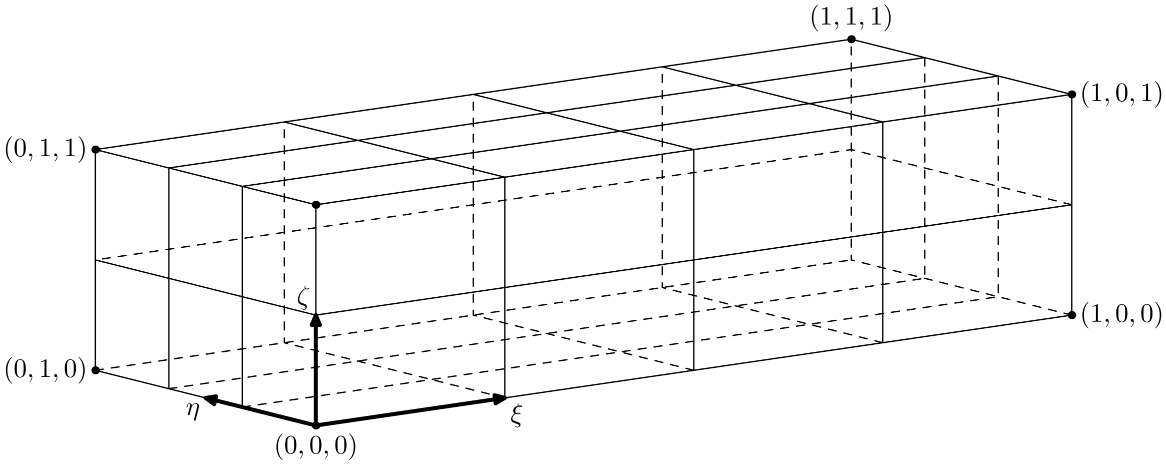

Figure G.51 – Parametric coordinate system for a 3-D structured mesh

EXPRESS specification:

```
*)
ENTITY structured_mesh
  SUBTYPE OF (mesh);
  vertex_counts : ARRAY [1:SELF\mesh.index_count] OF INTEGER;
  cell_counts   : ARRAY [1:SELF\mesh.index_count] OF INTEGER;
  rind_planes   : OPTIONAL rind;
END_ENTITY;
(*
```

Attribute definitions:

vertex_counts: The number of vertices in each dimension of the mesh. The product of the array elements is the number of vertices defining the mesh (i.e., excluding any rind points).

cell_counts: The number of cells in each dimension of the mesh. The product of the array elements is the number of cells on the interior of the mesh.

rind_planes: The rind planes associated with the mesh.

index_count: (inherited) The number of indices required to identify uniquely a vertex or cell in the mesh is the same as the topological dimensionality (e.g., 1–D, 3–D) of the mesh.

G.3.4.8 rind

rind describes the number of rind planes associated with a structured mesh.

EXPRESS specification:

```
*)
ENTITY rind;
  index_count : INTEGER;
  planes      : ARRAY [1:2*index_count] OF INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

index_count: The number of indices required to reference a vertex.

planes: contains the number of rind planes attached to the minimum and maximum faces of a structured mesh. The face corresponding to each index n of **planes** in 3-D is:

$$\begin{array}{ll} n = 1 \rightarrow i\text{-min} & n = 2 \rightarrow i\text{-max} \\ n = 3 \rightarrow j\text{-min} & n = 4 \rightarrow j\text{-max} \\ n = 5 \rightarrow k\text{-min} & n = 6 \rightarrow k\text{-max} \end{array}$$

EXAMPLE 1 For a 3-D grid whose 'core' size is `II`×`JJ`×`KK`, a value of **planes** `= [a,b,c,d,e,f]` indicates that the range of indices for the grid with this rind is:

```
i:   (1 - a, II + b)
j:   (1 - c, JJ + d)
k:   (1 - e, KK + f)
```

G.3.4.9 unstructured_mesh

An **unstructured_mesh** is a mesh where the topology need not be regular and the cell shapes are not restrained. It conceptually consists of the vertices of the mesh and the cells forming the volume of the mesh. The cells shall all be connected by each cell having at least one vertex in common with another cell. The shape of each cell in an unstructured mesh is explicitly specified.

EXPRESS specification:

```
*)
ENTITY unstructured_mesh
  SUBTYPE OF (mesh);
  cell_count   : INTEGER;
  cells        : ARRAY [1:cell_count] OF vertex_defined_cell;
END_ENTITY;
(*
```

Attribute definitions:

cell_count: is the number of cells in the mesh;

cells: is the cells forming the mesh.

G.3.4.10 vertex_defined_cell

A **vertex_defined_cell** is a region that is bounded by vertices; the number of vertices depends on the topological shape of the cell. The cell may have interior nodes; the maximum number of interior nodes depends on both the shape and the order of the cell.

EXPRESS specification:

```
*)
ENTITY vertex_defined_cell;
```

```
END_ENTITY;
(*
```

G.3.4.11 indices_list

indices_list specifies a list of indices into a multi-dimensional array.

EXPRESS specification:

```
*)
ENTITY indices_list;
  nindices : INTEGER;
  indices  : LIST [1:?] OF ARRAY [1:nindices] OF INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

nindices: the number of indices required to map to a unique array location;

indices: the indices.

G.3.4.12 indices_range

indices_range specifies the beginning and ending indices of a subrange in a multi-dimensional array.

EXPRESS specification:

```
*)
ENTITY indices_range;
  nindices : INTEGER;
  start    : ARRAY [1:nindices] OF INTEGER;
  finish   : ARRAY [1:nindices] OF INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

nindices: the number of indices required to map to a unique array location;

start: the indices of the minimal corner of the subrange;

finish: the indices of the maximal corner of the subrange.

G.3.4.13 grid_coordinates

The physical coordinates of the grid vertices in a zone are described by the **grid_coordinates** structure. The structure contains a list for the data arrays of the individual components of the position vector. It also provides a mechanism for identifying rind-point data included within the position-vector arrays.

EXPRESS specification:

```
*)
ENTITY grid_coordinates;
  descriptions : LIST OF TEXT;
  rind         : OPTIONAL rind;
  data         : LIST OF data_array;
END_ENTITY;
(*
```

Attribute definitions:

descriptions: is annotations;

rind: is optional. If not given then this is equivalent to a **rind** structure whose **planes** array contains all zeros.

data: is the grid-coordinate data; each **data_array** shall contain a single component of the position vector (e.g., three structures are required for 3-D data, one for each coordinate value).

Informal propositions:

ip1: Grid coordinates for an unstructured zone shall not have a value for **rind**, as it is meaningless in this case.

G.3.4.14 multiple_mesh_block

A **multiple_mesh_block** is a grouping of connected meshes. All mesh connectivity information pertaining to the group is contained in the **multiple_mesh_block** structure. This includes abutting interfaces (general mismatched and 1-to-1), overset-grid interfaces, and overset-grid holes.

All the interface patches for a given mesh in the group are contained in the **multiple_mesh_block** entity for that group. If a face of a mesh touches several other meshes (say N), the N different instances of the **mesh_connectivity** structure must be included in the **multiple_mesh_block** to describe each interface patch.

NOTE 1 This convention requires that a single interface patch be described twice — once for each adjacent mesh. It also means that the **multiple_mesh_block** is symmetrical with regard to interface patches.

EXPRESS specification:

```
*)
ENTITY multiple_mesh_block;
  annotation     : info;
  connectivities : LIST OF mesh_connectivity;
END_ENTITY;
(*
```

Attribute definitions:

annotation: is annotation;

connectivities: is the connectivity information.

G.3.4.15 mesh_connectivity

Information specifying the connectivity of a mesh interface.

All the interface patches for a given zone are contained in the **multiple_mesh_block** entity for that zone. If a face of a zone touches several other zones (say N), the N different instances of the **mesh_connectivity** structure must be included in the zone to describe each interface patch.

NOTE 1 This convention requires that a single interface patch be described twice in the database — once for each adjacent zone. It also means that the database is symmetrical with regard to interface patches.

EXPRESS specification:

```
*)
ENTITY mesh_connectivity;
  annotation : info;
  current    : mesh;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_mesh_connectivity FOR mesh_connectivity;
  ABSTRACT SUPERTYPE;
  ONEOF(matched_mesh_connection,
        mismatched_mesh_connection);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

annotation: is annotation;

current: is the current (receiver) mesh;

G.3.4.16 matched_mesh_connection

matched_mesh_connection contains connectivity information for a mesh interface patch that is abutting with 1-to-1 matching between adjacent structured mesh indices (also referred to as C0 connectivity). An interface patch is the subrange of the face of a mesh that touches one and only one other mesh. This structure identifies the subrange of indices for the two adjacent meshes that make up the interface and gives an index transformation from one mesh to the other.

The shorthand matrix notation used for the index transformation has the following properties. The matrix itself has rank **index_count** and contains elements +1, 0, and −1; it is orthonormal and its inverse is its transpose. The transformation matrix (`T`) works as follows: If `Index1` and

Index2 are the indices of a given point on the interface, where Index1 is in the current mesh and Index2 is in the adjacent mesh, then their relationship is,

```
Index2 = T.(Index1 - Start1) + Start2
Index1 = Transpose[T].(Index2 - Start2) + Start1
```

where the '.' notation indicates matrix-vector multiply, Start1 and Finish1 are the subrange indices contained in **range**, and Start2 and Finish2 are the subrange indices contained in **donor_range**.

The short-hand notation used for the **transform** is as follows. Each element shows the image in the adjacent mesh's face of a positive index increment in the current mesh's face. The first element is the image of a positive increment in i; the second element is the image of an increment in j; and the third (in 3-D) is the image of an increment in k in the current mesh's face. For 3-D, the transformation matrix T is constructed from **transform** $= [\pm a, \pm b, \pm c]$ as follows:

$$\mathtt{T} = \begin{bmatrix} \mathrm{sgn}(a)\mathrm{del}(a-1) & \mathrm{sgn}(b)\mathrm{del}(b-1) & \mathrm{sgn}(c)\mathrm{del}(c-1) \\ \mathrm{sgn}(a)\mathrm{del}(a-2) & \mathrm{sgn}(b)\mathrm{del}(b-2) & \mathrm{sgn}(c)\mathrm{del}(c-2) \\ \mathrm{sgn}(a)\mathrm{del}(a-3) & \mathrm{sgn}(b)\mathrm{del}(b-3) & \mathrm{sgn}(c)\mathrm{del}(c-3) \end{bmatrix},$$

where,

$$\mathrm{sgn}(x) \equiv \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \qquad \mathrm{del}(x-y) \equiv \begin{cases} 1, & \text{if } \mathrm{abs}(x) = \mathrm{abs}(y) \\ 0, & \text{otherwise} \end{cases}$$

EXAMPLE 1 **transform** = [-2, +3, +1] gives the transformation matrix,

$$\mathtt{T} = \begin{bmatrix} 0 & 0 & +1 \\ -1 & 0 & 0 \\ 0 & +1 & 0 \end{bmatrix}$$

NOTE 1 For establishing relationships between adjacent and current mesh indices lying on the interface itself, one of the elements of **transform** is superfluous since one component of both interface indices remains constant.

NOTE 2 The transform matrix and the two index pairs overspecify the interface patch. For example, Finish2 can be obtained from **transform**, Start1, Finish1 and Start2.

EXPRESS specification:

```
*)
ENTITY matched_mesh_connection
  SUBTYPE OF (mesh_connectivity);
  SELF\mesh_connectivity.current: structured_mesh;
  range       : indices_range;
  donor       : structured_mesh;
  donor_range : indices_range;
  transform   : ARRAY [1:index_count] OF INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

current: (inherited) is the current mesh;

index_count: (inherited) is the mesh dimension;

range: is the subrange of indices that define the interface patch in the current mesh;

donor: is the adjacent (donor) mesh.

donor_range: is the subrange of indices that define the interface patch in the donor mesh;

transform: is a shorthand notation for the transformation matrix describing the relationship between the indices of the two meshes.

G.3.4.17 mismatched_mesh_connection

mismatched_mesh_connection contains conectivity information for generalized mesh interfaces. Its purpose is to describe mismatched-abutting and overset interfaces for both structured and unstructured meshes, and can also be used for 1-to-1 abutting interfaces.

For abutting interfaces, also referred to as patched or mismatched, an interface patch is the subrange of the face of a mesh that touches one and only one other mesh. This structure identifies the subrange of indices (or array of indices) that make up the interface and gives their image in the adjacent (donor) mesh. It also identifies the adjacent mesh. If a given face of a mesh touches several (say N) adjacent meshes, then N different instances of **mismatched_mesh_connection** are needed to describe all the interfaces. For a single abutting interface, two instances of **mismatched_mesh_connection** are needed — one for each adjacent mesh.

For overset interfaces, this structure identifies the fringe points of a given mesh that lie in one and only one other mesh. If the fringe points of a mesh lie in several (say N) overlapping meshes, then N different instances of **mismatched_mesh_connection** are needed to describe the overlaps. It is possible with overset meshes that a single fringe point may actually lie in several overlapping meshes (though in typical usage, linkage to only one of the overlapping meshes is kept). There is no restriction against a given fringe point being contained within multiple instances of **mismatched_mesh_connection**; therefore, this structure allows the description of a single fringe point lying in several overlapping meshes.

EXPRESS specification:

```
*)
ENTITY mismatched_mesh_connection
  SUBTYPE OF (mesh_connectivity);
  points   : indices_group;
  gridloc  : mesh_location;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_mismatched_mesh_connection FOR
                   mismatched_mesh_connection;
  ABSTRACT SUPERTYPE;
  ONEOF(mismatched_mesh_region,
        mesh_overset_hole);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

current: (inherited) is the current mesh (the receiver mesh);

index_count: (inherited) is the number of indices required to reference a vertex.

points: is the indices of the interface points within the current mesh;

gridloc: is the location of indices within the current mesh described by **points**. It also identifies the location of indices described by an **index_range** in a donor mesh. This allows the flexibility to describe overset interfaces for cell-centered quantities.

G.3.4.18 mismatched_mesh_region

A mismatched connection that is abutting or overset.

EXPRESS specification:

```
*)
ENTITY mismatched_mesh_region
  SUBTYPE OF (mismatched_mesh_connection);
  donor : mismatched_donor_mesh;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_mismatched_mesh_region FOR mismatched_mesh_region;
  ABSTRACT SUPERTYPE;
  ONEOF(mesh_abutting,
        mesh_overset);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

donor: is an adjacent structured or unstructured donor mesh.

G.3.4.19 mesh_abutting

A mismatched abutting mesh connection.

EXPRESS specification:

```
*)
ENTITY mesh_abutting
  SUBTYPE OF (mismatched_mesh_region);
END_ENTITY;
(*
```

Informal propositions:

ip1: The **range** or **vertices** shall describe a face subrange (i.e., points in a single computational grid plane);

ip2: The **structured_donor** shall also describe a face subrange;

G.3.4.20 mesh_overset

An overset mesh connection.

EXPRESS specification:

```
*)
ENTITY mesh_overset
  SUBTYPE OF (mismatched_mesh_region);
END_ENTITY;
(*
```

G.3.4.21 mesh_overset_hole

Grid connectivity for overset meshes may also include 'holes' within meshes, where any mesh data is ignored or 'turned off', because the data in some other overlapping mesh applies instead. **overset_hole** specifies those points within a given mesh that make up a hole (or holes).

EXPRESS specification:

```
*)
ENTITY mesh_overset_hole
  SUBTYPE OF (mismatched_mesh_connection);
END_ENTITY;
(*
```

NOTE 1 The interface points making up a hole within a mesh may be specified by an element in the **range** list if they constitute a logically rectangular region. Likewise further elements in the list may be used for further logically rectangular holes. The more general alternative is to use **vertices** to list all interface points making up the holes within a mesh. Using the list of **range** specifications, or using **range** in combination with **vertices**, may result in a given hole being specified more than once.

G.3.4.22 mismatched_donor_mesh

A **mismatched_donor_mesh** is a mesh that acts as a donor for a **mismatched_mesh_region**.

EXPRESS specification:

```
*)
ENTITY mismatched_donor_mesh;
INVERSE
  connect : mismatched_mesh_region FOR donor;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_mismatched_donor_mesh FOR
                   mismatched_donor_mesh;
  ABSTRACT SUPERTYPE;
  ONEOF(structured_donor_mesh,
        unstructured_donor_mesh);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

connect: is the **mismatched_mesh_region** for which this is the **donor**.

G.3.4.23 structured_donor_mesh

A **structured_donor_mesh** is a **mismatched_donor_mesh** that is structured.

EXPRESS specification:

```
*)
ENTITY structured_donor_mesh
  SUBTYPE OF (mismatched_donor_mesh);
  donor  : structured_mesh;
  points : data_array;
  vsize  : INTEGER;
DERIVE
  index_count : INTEGER := donor.index_count;
END_ENTITY;
(*
```

Attribute definitions:

donor: is the structured donor mesh;

points: is the image of the receiver mesh interface points in the donor mesh. These may be thought of as bi- or tri-linear interpolants (depending on **dimension**) in the computational grid of the donor mesh. FORTRAN multidimensional array ordering shall be used;

vsize: is the size of the data array necessary to contain the interface points;

index_count: is the number of indices required to reference a vertex.

G.3.4.24 unstructured_donor_mesh

An **unstructured_donor_mesh** is a **mismatched_donor_mesh** that is unstructured.

EXPRESS specification:

```
*)
ENTITY unstructured_donor_mesh
  SUBTYPE OF (mismatched_donor_mesh);
  donor       : unstructured_mesh;
  cells       : indices_group;
  interpolant : data_array;
  vsize       : INTEGER;
DERIVE
  index_count : INTEGER := donor.index_count;
  cell_dim    : INTEGER := donor.cell_dim;
END_ENTITY;
(*
```

Attribute definitions:

donor: is the unstructured donor mesh;

cells: contains the donor cell where the node is located.

interpolants: contains the interpolation factors to locate the node in the donor cell.

vsize: is the size of the data array necessary;

index_count: is the number of indices required to reference a vertex;

cell_dim: is the dimension of a cell in the mesh.

G.3.4.25 condition

A **condition** represents the concept of conditions or constraints pertinent to to an analysis.

EXPRESS specification:

```
*)
ENTITY condition;
--  SUBTYPE OF (state_property_distribution);
  annotation : info;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_condition FOR condition;
  ABSTRACT SUPERTYPE;
  ONEOF(zone_bc,
        boundary_condition,
        bc_dataset,
        bc_data,
        reference_state,
        integral_data);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

annotation: is annotation.

G.3.4.26 zone_bc

All boundary-condition information pertaining to a given zone is contained in the **zone_bc** structure.

EXPRESS specification:

```
*)
ENTITY zone_bc
  SUBTYPE OF (condition);
  conditions : LIST OF boundary_condition;
  rstate     : OPTIONAL reference_state;
```

```
END_ENTITY;
(*
```

Attribute definitions:

conditions: is the boundary-conditions for a zone, on a patch by patch basis. Boundary-condition information for a single patch is contained in the **bc** structure. If a zone contains *N* boundary-condition patches, then *N* separate instances of **bc** shall be provided in the **zone_bc** entity for the zone.

rstate: is non-default reference data. Reference data is applicable to all the boundary-condition of the zone. Reference state data is useful for situations where boundary-condition data is not provided, and solvers are free to enforce any appropriate boundary-condition equations.

EXAMPLE An engine nozzle exit boundary-condition usually imposes a stagnation pressure (or some other stagnation quantity) different from freestream. The nozzle-exit stagnation quantities could be specified by reference data at this level or below in lieu of providing explicit Dirichlet or Neumman data.

G.3.4.27 boundary_condition

boundary_condition contains boundary-condition information for a single BC surface patch of a zone. A BC patch is the subrange of the face of a zone where a given boundary-condition is applied.

The structure contains a boundary-condition type, as well as one or more sets of boundary-condition data that are used to define the boundary-condition equations to be enforced on the BC patch. For most boundary-conditions, a single data set is all that is needed. The structure also contains information describing the normal vector to the BC surface patch.

EXPRESS specification:

```
*)
ENTITY boundary_condition
  SUBTYPE OF (condition);
  the_type            : bc_type;
  gridloc             : mesh_location;
  inward_normal_index : OPTIONAL ijk_minmax;
  inward_normal_list  : OPTIONAL indices_list;
  datasets            : LIST OF bc_dataset;
  family              : OPTIONAL family;
  rstate              : OPTIONAL reference_state;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_boundary_condition FOR boundary_condition;
  ONEOF(elements_bc,
        element_list_bc,
        element_range_bc,
        point_list_bc,
        point_range_bc);
END_SUBTYPE_CONSTRAINT;
(*
```

Table G.8 – inward_normal_index values

Face	Value	Face	Value
i_min	$[+1, 0, 0]$	**i_max**	$[-1, 0, 0]$
j_min	$[0, +1, 0]$	**j_max**	$[0, -1, 0]$
k_min	$[0, 0, +1]$	**k_max**	$[0, 0, -1]$

Attribute definitions:

the_type: is the type of the boundary-condition;

gridloc: is the location of the conditions, which are normally either at the cell vertices or on the cell faces.

inward_normal_index: indicates the computational-coordinate direction of the BC patch normal; this normal points into the interior of the zone.

Some boundary-conditions require a normal direction to be specified in order to be properly imposed. A computational-coordinate normal can be derived from **point_range** or **point_list** by examining redundant index components. Alternatively, this information can be provided directly by **inward_normal_index**. For exterior faces of a zone in 3-D, **inward_normal_index** takes one of the values given in Table G.8.

inward_normal_list: is a list of vectors normal to the BC patch pointing into the interior of the zone; the vectors are not required to be unit vectors. By convention the vectors are located at the vertices of the BC patch.

The physical-space normal vectors of the BC patch may be described by **inward_normal_list**; these are located at vertices, consistent with **point_range** and **point_list**. **inward_normal_list** is specified as an optional attribute because it is not always needed to enforce boundary-conditions, and the physical-space normals of a BC patch can usually be constructed from the grid. However, there are some situations, such as grid-coordinate singularity lines, where **inward_normal_list** becomes a required attribute because the normals cannot be generated from other information.

datasets: is a list of boundary-condition data sets. In general, the proper **bc_dataset** instance(s) to impose on the BC patch is determined by the value of **the_type**.

For a few boundary-conditions, such as a symmetry plane or polar singularity, the value of **the_type** completely describes the equations to impose, and no instances of **bc_dataset** are needed. For ‘simple’ boundary-conditions, where a single set of Dirichlet and/or Neumann data is applied a single **bc_dataset** will likely be used (although this is not a requirement). For ‘compound’ boundary-conditions, where the equations to impose are dependent on local conditions, several instances of **bc_dataset** will likely be used.

rstate: is non-default reference data. Reference data is applicable to the conditions of the BC patch.

G.3.4.28 elements_bc

An **elements_bc** is an **boundary_condition** containing boundary-condition information for a group of elements in an unstructured mesh.

EXPRESS specification:

```
*)
ENTITY elements_bc
  SUBTYPE OF (boundary_condition);
  elements : LIST OF vertex_defined_cell;
END_ENTITY;
(*
```

Attribute definitions:

elements: is the elements (in an unstructured mesh).

G.3.4.29 element_list_bc

An **element_list_bc** is an **boundary_condition** containing boundary-condition information for a list of cells in a structured mesh.

EXPRESS specification:

```
*)
ENTITY element_list_bc
  SUBTYPE OF (boundary_condition);
  element_list : indices_list;
END_ENTITY;
(*
```

Attribute definitions:

element_list: is the element indices;

G.3.4.30 element_range_bc

An **element_range_bc** is an **boundary_condition** containing boundary-condition information for a range of cells in a structured mesh.

EXPRESS specification:

```
*)
ENTITY element_range_bc
  SUBTYPE OF (boundary_condition);
  element_range : indices_range;
END_ENTITY;
(*
```

Attribute definitions:

element_range: is an element subrange;

G.3.4.31 point_list_bc

A **point_list_bc** is an **boundary_condition** containing boundary-condition information for a list of points in a structured mesh.

EXPRESS specification:

```
*)
ENTITY point_list_bc
  SUBTYPE OF (boundary_condition);
  point_list              : indices_list;
  vertex_list_length      : INTEGER;
  face_center_list_length : INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

point_list: is a list of points; by convention the indices refer to vertices;

vertex_list_length: is the number of vertices making up the BC patch.

face_center_list_length: is the number of cell faces making up the BC patch.

G.3.4.32 point_range_bc

A **point_range_bc** is an **boundary_condition** containing boundary-condition information for a range of points in a structured mesh.

EXPRESS specification:

```
*)
ENTITY point_range_bc
  SUBTYPE OF (boundary_condition);
  point_range : indices_range;
END_ENTITY;
(*
```

Attribute definitions:

point_range: is a face subrange (i.e., points in a single computational plane); by convention the indices refer to vertices;

Informal propositions:

ip1: If **inward_normal_list** is specified, then a an ordering convention is needed for indices on the BC patch. An ordering convention is also needed if local data is present in the **bc_dataset** substructures. FORTRAN multidimensional array ordering shall be used.

G.3.4.33 bc_dataset

bc_dataset contains Dirichlet and Neumann data for a single set of boundary-condition equations. Its intended use is for simple boundary-condition types, where the equations imposed do not depend on local conditions.

Boundary-condition data is separated by equation type into Dirichlet and Neumann conditions. Dirichlet boundary-conditions impose the value of the given variables, whereas Neumann boundary-conditions impose the normal derivative of the given variables.

The **boundary_condition** structure (clause G.3.4.27) allows for an arbitrary list of boundary-condition data sets, described by the **bc_dataset** structure. For simple boundary-conditions, a single data set must be chosen from a list that may contain more than one element. Likewise, for a compound boundary-condition, a limited number of data sets must be chosen and applied appropriately. The mechanism for proper choice of data sets can be controlled by the **the_type** attribute of the **boundary_condition** structure together with the **the_type** attribute of the **bc_dataset** structure, and the boundary-condition type association table (Table G.9).

boundary_condition is used for both simple and compound boundary-conditions; hence, its attribute **the_type** is of type **bc_type**. Conversely, the structure **bc_dataset** is intended to enforce a single set of boundary-condition equations independent of local flow conditions (i.e., it is appropriate only for simple boundary-conditions). That is why its attribute **the_type** is of type **bc_type_simple** and not **bc_type**.

NOTE 1 The appropriate choice of data sets may be specified by listing appropriate pairings of the values of **boundary_condition.the_type** and **bc_dataset.the_type**.

Although the model has a strict division between the two categories of boundary-condition types, in practice some overlap may exist. These complications require further guidelines on appropriate definition and use of boundary-condition types. The real distinctions between **bc_type_simple** and **bc_type_compound** are as follows:

— **bc_type_simple** identifiers always match themselves; **bc_type_compound** never match themselves.

— **bc_type_simple** identifiers always produce a single match; **bc_type_compound** produce multiple matches.

— The usage rule for **bc_type_simple** identifiers is always trivial — apply the single matching data set regardless of local conditions.

Therefore, any boundary-condition that involves application of different data sets depending on local conditions should be classified as **bc_type_compound**.

NOTE 2 If a type that is classified **bc_type_simple** is desired to be used as a compound, somehow be reclassified. One option is to define a new **bc_type_compound** identifier and provide associated **bc_type_simple** types and a usage rule. Another option may be to allow some identifiers to be both **bc_type_simple** and **bc_type_compound** and let their appropriate use be based on context.

For a given simple boundary-condition the database provides a set of boundary-condition equations to be enforced through the definitions of **bc_dataset** and **bc_data**. Apart from the boundary-condition type, the precise equations to be enforced are described by boundary-condition solution data. These specified solution data are arranged by 'equation type':

Table G.9 – Associated boundary-condition types and usage rules

bc_type Identifier	**Associated bc_type_simple identifiers and usage rules**
bc_inflow	**bc_inflow_supersonic** **bc_inflow_subsonic** *usage rule:* if supersonic normal Mach, choose **bc_inflow_supersonic**, else choose **bc_inflow_subsonic**.
bc_Outflow	**bc_outflow_supersonic** **bc_outflow_subsonic** *usage rule:* if supersonic normal Mach, choose **bc_outflow_supersonic**, else choose **bc_outflow_subsonic**.
bc_farfield	**bc_inflow_supersonic** **bc_inflow_subsonic** **bc_outflow_supersonic** **bc_outflow_subsonic** *usage rule:* if inflow and supersonic normal Mach, choose **bc_inflow_-supersonic**, else if inflow, choose **bc_inflow_subsonic**, else if outflow and supersonic normal Mach, choose **bc_-outflow_supersonic**, else, choose **bc_outflow_subsonic**.
bc_inflow_supersonic	**bc_inflow_supersonic** **bc_Dirichlet** *usage rule:* choose either; **bc_inflow_supersonic** takes precedence.
bc_outflow_supersonic	**bc_outflow_supersonic** **bc_Extrapolate** *usage rule:* choose either; **bc_outflow_supersonic** takes precedence.
all others	self-matching

Dirichlet: $Q = (Q)_{\text{specified}}$
Neumann: $\partial Q/\partial n = (\partial Q/\partial n)_{\text{specified}}$

The **Dirichlet_data** and **Neumann_data** attributes (of type **bc_data**) list both the solution variables involved in the equations (through data-name conventions) and the specified solution data.

Two issues need to be addressed for specifying Dirichlet or Neumann boundary-condition data. The first is whether the data is global or local.

The global versus local issue can easily be handled by storing a scalar for the global BC data case, and storing an array for the local BC data case.

By convention, if the **Dirichlet_data** and **Neumann_data** are not present in an instance of

bc_dataset, then application codes are free to enforce appropriate boundary-conditions for the given type of **bc_type_simple**. Furthermore, if insufficient data is present then application codes are free to fill out the boundary-condition data as appropriate for the **bc_type_simple** identifier.

To facilitate implementation of boundary-conditions into existing solvers, if no boundary-condition data is specified, solvers are free to enforce any appropriate boundary-condition equations. This includes situations where instances of **bc_dataset**, **bc_data** or **data_array** are absent within the boundary-condition hierarchy. In this case the **reference_state** specifies the reference-state conditions from which the solver should extract the boundary-condition data. Within the boundary-condition hierarchy, **reference_state** instances may be present at any of the **zone_bc**, **boundary_condition** or **bc_dataset** levels with the lowest taking precedence.

EXPRESS specification:

```
*)
ENTITY bc_dataset
  SUBTYPE OF (condition);
  the_type       : bc_type_simple;
  gridloc        : OPTIONAL mesh_location;
  rstate         : OPTIONAL reference_state;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_bc_dataset FOR bc_dataset;
  ONEOF(Dirichlet_bc_dataset,
        Neumann_bc_dataset);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

the_type: is the boundary-condition type, which gives general information on the boundary-condition equations to be enforced.

gridloc: is the location of local data arrays (if any) provided in **Dirichlet_bc_dataset** or **Neumann_bc_dataset**. Local boundary-condition data may be defined either at vertices or boundary face centers.

rstate: is non-default reference data. Reference data is applicable to the set of boundary-condition data;

G.3.4.34 Dirichlet_bc_dataset

An **Dirichlet_bc_dataset** is an **bc_dataset** that contains Dirichlet boundary condition data.

EXPRESS specification:

```
*)
ENTITY Dirichlet_bc_dataset
  SUBTYPE OF (bc_dataset);
  Dirichlet_data : bc_data;
END_ENTITY;
```

```
(*
```

Attribute definitions:

Dirichlet_data: is boundary-condition data for Dirichlet conditions which may be constant over the BC patch or defined locally at each point of the patch.

G.3.4.35 Neumann_bc_dataset

An **Neumann_bc_dataset** is an **bc_dataset** that contains Neumann boundary condition data.

EXPRESS specification:

```
*)
ENTITY Neumann_bc_dataset
  SUBTYPE OF (bc_dataset);
  Neumann_data : bc_data;
END_ENTITY;
(*
```

Attribute definitions:

Neumann_data: is boundary-condition data for Neumann conditions which may be constant over the BC patch or defined locally at each point of the patch.

G.3.4.36 bc_data

bc_data contains a list of variables and associated data for boundary-condition specification. Each variable may be given as global data (i.e., a scalar) or local data defined at each grid point of the BC patch. By convention all data specified in a given instance of **bc_data** is to be used in the same *type* of boundary-condition equation.

EXAMPLE 1 The Dirichlet and Neumann conditions in **bc_dataset** use separate **bc_data** structures.

This structure allows a given instance of **bc_data** to have a mixture of global and local data.

EXAMPLE 2 If the Dirichlet condition consists of a uniform stagnation pressure but with with a non-uniform velocity profile, then the stagnation pressure can be described by a scalar in the **data_global** list and the velocity by an array in the **data_local** list.

EXPRESS specification:

```
*)
ENTITY bc_data
  SUBTYPE OF (condition);
  data_global : LIST OF data_array;
  data_local  : LIST OF data_array;
END_ENTITY;
```

Table G.10 – Data that may be associated with reference_state

data_name	Description	Units
Mach	Mach number, $M = q/c$	—
Mach_velocity	Velocity scale, q	**L/T**
Mach_velocity_sound	Speed of sound scale, c	**L/T**
Reynolds	Reynolds number, $Re = VL_R/\nu$	—
Reynods_velocity	Velocity scale, V	**L/T**
Reynolds_length	Length scale, L_R	**L**
Reynolds_viscosity_kinematic	Kinematic viscosity scale, ν	$\mathbf{L}^2$**/T**
length_reference	Reference length, L_{ref}	**L**

```
(*
```

Attribute definitions:

data_global: is global data;

data_local: is local data;

G.3.4.37 reference_state

reference_state describes a reference state, which is a list of geometric or state quantities defined at a common location or condition.

EXAMPLE Typical reference states associated with CFD calculations are freestream, plenum, stagnation, inlet and exit.

The data that may be associated with **reference_state** is listed in Table G.10. The meaning of the identifiers is described in clause G.3.3.6.

EXPRESS specification:

```
*)
ENTITY reference_state
  SUBTYPE OF (condition);
  state_description : LIST OF STRING;
  data              : LIST OF data_array;
END_ENTITY;
(*
```

Attribute definitions:

state_description: is a human interpretable description of the reference state;

data: is the reference state data.

G.3.4.38 integral_data

integral_data provides a description of generic global or integral data that may be associated with a particular zone or an entire database. In contrast to **discrete_data**, integral data is not associated with any specific field location.

EXPRESS specification:

```
*)
ENTITY integral_data
  SUBTYPE OF (condition);
  data : LIST OF data_array;
END_ENTITY;
(*
```

Attribute definitions:

data: is the data;

G.3.4.39 family

EXPRESS specification:

```
*)
ENTITY family;
  annotation : info;
  conditions : LIST OF bc_type;
  geometry   : LIST OF geometry_reference;
END_ENTITY;
(*
```

Attribute definitions:

annotation: is annotation;

conditions: the family's boundary conditions;

geometry: the family's geometric information;

G.3.4.40 equation

A **equation** represents the concept of a mathematical formulation of a physics phenonema.

EXPRESS specification:

```
*)
ENTITY equation;
--  SUBTYPE OF (behaviour);
  annotation : info;
END_ENTITY;
```

```
SUBTYPE_CONSTRAINT sc1_equation FOR equation;
  ONEOF(equation_set,
        governing_equation,
        behaviour_model);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

annotation: is annotation.

G.3.4.41 equation_set

An **equation_set** is a **equation** that gives a general description of governing flow equations. It includes the dimensionality of the governing equations.

EXPRESS specification:

```
*)
ENTITY equation_set
  SUBTYPE OF (equation);
  dimension             : INTEGER;
  equations             : governing_equation;
  dclass                : OPTIONAL data_class;
  dimunits              : OPTIONAL dimensional_units;
  models                : SET OF behaviour_model;
END_ENTITY;
(*
```

Attribute definitions:

dimension: is the dimensionality of the governing equations; it is the number of spatial variables describing the equation;

equations: describes the general class of equations.

dclass: non-default class of data;

dimunits: non-default system of units;

dclass: is the non-default class of data contained in the **equation_set**.

models: describes zero or more of:

— the gaseous equation of state;

— the auxiliary relations for molecular viscosity;

— the turbulent closure for Reynolds-averaged Navier-Stokes equations;

— the turbulence model for Reynolds-averaged Navier-Stokes equations.

G.3.4.42 governing_equation

governing_equation is an **equation** describing the class of governing flow equations associated with the analysis solution.

EXPRESS specification:

```
*)
ENTITY governing_equation
  SUBTYPE OF (equation);
  equation_type : governing_equation_type;
END_ENTITY;
(*
```

Attribute definitions:

equation_type: is the kind of equation;

G.3.4.43 diffusion_equation

A **diffusion_equation** is a **governing_equation** which includes diffusion.

EXPRESS specification:

```
*)
ENTITY diffusion_equation
  SUBTYPE OF (governing_equation);
  diffusion_model : diffusion_model;
END_ENTITY;
(*
```

Attribute definitions:

diffusion_model: describes the viscous diffusion terms modelled in the flow equations, and is applicable only to Navier-Stokes equations.

G.3.4.44 behaviour_model

An **behaviour_model** describes a model for relating physical or mathematical quantities.

EXPRESS specification:

```
*)
ENTITY behaviour_model
  SUBTYPE OF (equation);
  data     : LIST OF data_array;
  dclass   : OPTIONAL data_class;
  dimunits : OPTIONAL dimensional_units;
```

Table G.11 – Encoding of the 3-D diffusion_model terms

Element	Modelled terms
$n = 1$	diffusion terms in i ($\partial^2/\partial\xi^2$)
$n = 2$	diffusion terms in j ($\partial^2/\partial\eta^2$)
$n = 3$	diffusion terms in k ($\partial^2/\partial\zeta^2$)
$n = 4$	cross-diffusion terms in i-j ($\partial^2/\partial\xi\partial\eta$ and $\partial^2/\partial\eta\partial\xi$)
$n = 5$	cross-diffusion terms in j-k ($\partial^2/\partial\eta\partial\zeta$ and $\partial^2/\partial\zeta\partial\eta$)
$n = 6$	cross-diffusion terms in k-i ($\partial^2/\partial\zeta\partial\xi$ and $\partial^2/\partial\xi\partial\zeta$)

```
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_behaviour_model FOR behaviour_model;
  ABSTRACT SUPERTYPE;
  ONEOF(thermal_conductivity_model,
        gas_model,
        turbulence_closure,
        turbulence_model,
        viscosity_model);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

data: is the data;

dclass: is non-default class of data;

dimunits: is non-default dimensional units.

G.3.4.45 diffusion_model

The **diffusion_model** describes the viscous diffusion terms modelled in the flow equations, and is applicable only to Navier-Stokes equations.

Typically, thin-layer approximations include only the diffusion terms in one or two computational-coordinate directions. **diffusion_model** encodes the coordinate directions that include second-derivative and cross-derivative diffusion terms. The first N elements, where N is the computational dimension, are second-derivative terms and the remainder elements are cross-derivative terms. A value of TRUE indicates the diffusion term is modelled, and FALSE indicates that it is not modelled. In 3–D, the encoding of the **diffusion_model** terms is given in Table G.11, where derivatives in the i, j and k computational-coordinates are ξ, η and ζ, respectively.

EXAMPLE The full Navier-Stokes equations in 3–D are indicated by:
terms = [TRUE,TRUE,TRUE,TRUE,TRUE,TRUE]
while the thin-layer equations including only diffusion in the j-direction are indicated by:
terms = [FALSE,TRUE,FALSE,FALSE,FALSE,FALSE].

EXPRESS specification:

```
*)
ENTITY diffusion_model;
```

```
  terms : ARRAY [1:diff] OF BOOLEAN;
  diff  : INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

terms: is the diffusion terms;

diff: is the number of elements in the **terms** array. For 1-D this is one, for 2-D it is three, and for 3-D it is six.

G.3.4.46 gas_model

gas_model describes the equation of state model used in the governing equations to relate pressure, temperature and density.

EXPRESS specification:

```
*)
ENTITY gas_model
  SUBTYPE OF (behaviour_model);
  model_type : gas_model_type;
END_ENTITY;
(*
```

Attribute definitions:

model_type: is the particular gaseous equation of state model.

G.3.4.47 thermal_conductivity_model

A **thermal_conductivity_model** describes the model for relating the thermal-conductivity coefficient (k) to temperature.

EXPRESS specification:

```
*)
ENTITY thermal_conductivity_model
  SUBTYPE OF (behaviour_model);
  model_type : thermal_conductivity_model_type;
END_ENTITY;
(*
```

Attribute definitions:

model_type: is the particular thermal conductivity model type.

G.3.4.48 turbulence_closure

turbulence_closure describes the turbulence closure for the Reynolds stress terms of the Navier-Stokes equations.

EXPRESS specification:

```
*)
ENTITY turbulence_closure
  SUBTYPE OF (behaviour_model);
  closure_type : turbulence_closure_type;
END_ENTITY;
(*
```

Attribute definitions:

closure_type: is the particular turbulence closure type.

G.3.4.49 turbulence_model

turbulence_model describes the equation set used to model the turbulence quantities.

EXPRESS specification:

```
*)
ENTITY turbulence_model
  SUBTYPE OF (behaviour_model);
  model_type      : turbulence_model_type;
  diffusion_model : OPTIONAL diffusion_model;
END_ENTITY;
(*
```

Attribute definitions:

model_type: is the particular turbulence model type;

diffusion_model: is the description of the viscous diffusion terms included in the turbulent transport model equations.

G.3.4.50 viscosity_model

viscosity_model describes the model for relating molecular viscosity (μ) to temperature.

EXPRESS specification:

```
*)
ENTITY viscosity_model
  SUBTYPE OF (behaviour_model);
  model_type : viscosity_model_type;
END_ENTITY;
(*
```

Attribute definitions:

model_type: is the particular viscosity model type.

G.3.4.51 result

An **result** represents the concept of a solution to an analysis problem and/or other data resulting from an analysis.

EXPRESS specification:

```
*)
ENTITY result;
--  SUBTYPE OF (state_property_distribution);
  annotation : info;
END_ENTITY;

SUBTYPE_CONSTRAINT sc1_result FOR result;
  ONEOF(solution,
        convergence_history,
        discrete_data);
END_SUBTYPE_CONSTRAINT;
(*
```

Attribute definitions:

annotation: is annotation.

G.3.4.52 solution

The solution within a given zone is described by the **solution** structure. This structure contains a list of the data arrays of the individual flow solution variables, as well as identifying the grid location of the solution. There is a mechanism for identifying rind-point data included within the data arrays.

EXPRESS specification:

```
*)
ENTITY solution
  SUBTYPE OF (result);
  solution : LIST OF data_array;
  gridloc  : mesh_location;
END_ENTITY;
(*
```

Attribute definitions:

solution: is the solution data. Each structure in the list contains a single component of the solution vector.

gridloc: specifies the location of the solution data with respect to the grid. All data within a given instance of **solution** resides at the same kind of grid location.

G.3.4.53 solution_with_rind

An **solution_with_rind** is an **solution** with rind point data.

EXPRESS specification:

```
*)
ENTITY solution_with_rind
  SUBTYPE OF (solution);
  rind_planes : rind;
END_ENTITY;
(*
```

Attribute definitions:

rind_planes: is the number of rind planes included in the data.

Informal propositions:

ip1: A solution of an unstructured zone shall not have a value for **rind**, as it is meaningless in this case.

G.3.4.54 convergence_history

Solver convergence history information is described by the **convergence_history** structure.

Measures used to record convergence vary greatly among current solver implementations. Convergence information typically includes global forces, norms of equation residuals, and norms of solution changes.

NOTE 1 Attempts to systematically define a set of convergence measures have been futile. For global parameters, such as forces and moments (Table G.4), a set of standarized data-array identifiers can be agreed. For equation residuals and solution changes, no such standard list exists. Therefore, either the **id** attribute of the **general_property** associated with the **data_array**s, or an **externally_defined_item** value for the **specified_name** of an associated **specified_general_property** has to be used as the data-array identifier. It is suggested that identifiers for norms of equation residuals begin with RSD, and those for solution changes begin with CHG. For example, **'RSD Mass RMS'** could be used for the L_2-norm (RMS) of mass conservation residuals.

EXPRESS specification:

```
*)
```

```
ENTITY convergence_history
  SUBTYPE OF (result);
  norm_definitions : STRING;
  iterations       : INTEGER;
  data             : LIST OF data_array;
END_ENTITY;
(*
```

Attribute definitions:

norm_definitions: is a description of the convergence information recorded as **data**;

iterations: is the number of iterations for which convergence information is recorded;

data: is convergence history data;

G.3.4.55 discrete_data

discrete_data provides a description of generic discrete data (i.e., data defined on a computational grid); it is identical to **solution** except for its entity name. This structure can be used to store field data, such as fluxes or equation residuals, that is not typically considered part of the solution.

EXPRESS specification:

```
*)
ENTITY discrete_data
  SUBTYPE OF (result);
  data    : LIST OF data_array;
  gridloc : mesh_location;
END_ENTITY;
(*
```

Attribute definitions:

data: is the data;

gridloc: is the location of the data with respect to the grid. All data within a given instance of **discrete_data** resides at the same kind of grid location.

G.3.4.56 discrete_data_with_rind

An **discrete_data_with_rind** is an **discrete_data** with rind point data.

EXPRESS specification:

```
*)
ENTITY discrete_data_with_rind
  SUBTYPE OF (discrete_data);
  rind_planes : rind;
END_ENTITY;
```

```
(*
```

Attribute definitions:

rind_planes: is the number of rind planes included in the data.

G.3.4.57 data_conversion

data_conversion contains conversion factors for recovering raw dimensional data from given nondimensional data.

Given a nondimensional piece of data, `Data(nondimensional)`, the conversion to 'raw' dimensional form is:

```
Data(raw) = Data(nondimensional)*scale + offset
```

EXPRESS specification:

```
*)
ENTITY data_conversion;
  scale  : REAL;
  offset : REAL;
END_ENTITY;
(*
```

Attribute definitions:

scale: The scaling factor.

offset: The offset.

G.3.4.58 dimensional_units

dimensional_units describes the system of fundamental units used to measure dimensional data.

EXPRESS specification:

```
*)
ENTITY dimensional_units;
END_ENTITY;
-- map to global_unit_assigned_context (measure_schema)
(*
```

G.3.4.59 dimensional_exponents

dimensional_exponents contains the dimensional units exponents.

EXPRESS specification:

```
*)
ENTITY dimensional_exponents;
END_ENTITY;
-- map to dimensional_exponents (measure_schema)
(*
```

G.3.4.60 data_array

data_array describes a multi-dimensional data array of a given type, dimensionality and size in each dimension. The data may be dimensional, nondimensional or pure constants. Qualifiers are provided to describe dimensional units or normalization information associated with the data.

This structure is formulated to describe an array of scalars. Therefore, for vector quantities (e.g., a position vector or a velocity vector), separate instances are required for each component of the vector.

EXAMPLE 1 The cartesian coordinates of a 3-D grid are described by three separate data arrays: one for x, one for y, and one for z.

The *name* for the data is specified by the associated **data_name**.

The *class* of the data is specified by the associated **data_class**.

The optional attributes of **data_array** provide information for manipulating the data, including changing units or normalization. Within a given instance of **data_array** where the data class is specified by a **defined_data_class**, all information required for manipulations may be completely and precisely specified by the data class and the values of **units**, **exponents** and **conversion**.

— **dimensional:** When the data class is **dimensional**, the data is dimensional. The optional qualifiers **units** and **exponents** describe dimensional units associated with the data. These qualifiers are provided to specify the system of dimensional units and the dimensional exponents, respectively.

  EXAMPLE 2 If the data is the x-component of velocity, then **units** will state that the pertinent dimensional units are, say, **metre** and **second**; **exponents** will specify that the pertinent dimensional exponents are **length** = 1 and **time** = -1. Combining the information gives the units *m/s*.

  If **exponents** is absent, then the appropriate dimensional exponents can determined by convention provided the data name is one of a set of standard identifiers, otherwise the exponents are unspecified.

— **normalized_by_dimensional:** When the data class is **normalized_by_dimensional**, the data is nondimensional and is normalized by dimensional reference quantities. All optional entities in **data_array** are used. **conversion** contains factors to convert the nondimensional data to 'raw' dimensional data; these factors are **scale** and **offset**. The conversion process is as follows:

```
Data(raw) = Data(nondimensional)*scale + offset
```

where `Data(nondimensional)` is the original nondimensional data, and `Data(raw)` is the converted raw data. This converted raw data is dimensional, and the qualifiers **units** and **exponents** describe the appropriate dimensional units and exponents. Note that **units** and **exponents** also describe the units for **scale** and **offset**.

If **conversion** is absent, the equivalent defaults are **scale** = 1 and **offset** = 0. If either **units** or **exponents** is absent, follow the rules described for **dimensional** data above.

— **normalized_by_unknown_dimensional:** When the data class is **normalized_by_unknown_dimensional**, the data is nondimensional and is normalized by some unspecified dimensional quantities. This type of data is typical of a completely nondimensional test case, where all field data and all reference quantities are nondimensional.

Only the **exponents** qualifier is used in this case, although it is expected that this qualifier will be seldom utilized in practice. For entities of **data_array** that are not among the list of standardized data-name identifiers, the qualifier could provide useful information by defining the exponents of the dimensional form of the nondimensional data.

Rather than providing qualifiers to describe the normalization of the data, all data of type **normalized_by_unknown_dimensional** in a given object base shall be nondimensionalized consistently. This is done by picking one set of mass, length, time and temperature scales and normalizing all appropriate data by these scales. This process is described in detail in the following.

NOTE 1 The practice of nondimensionalization within flow solvers and other application codes is quite popular. The problem with this practice is that to manipulate the data from a given code, one must often know the particulars of the nondimensionalization used. This largely results from what can be termed inconsistent normalization — more than the minimum required scales are used to normalize data within the code.

EXAMPLE 3 In one CFD flow solver, the following nondimensionalization is used:

$$\begin{array}{ccc} \tilde{x} = x/L, & \tilde{u} = u/c_\infty, & \tilde{\rho} = \rho/\rho_\infty, \\ \tilde{y} = y/L, & \tilde{v} = v/c_\infty, & \tilde{p} = p/(\rho_\infty c_\infty^2), \\ \tilde{z} = z/L, & \tilde{w} = w/c_\infty, & \tilde{\mu} = \mu/\mu_\infty, \end{array}$$

where (x, y, z) are the cartesian coordinates, (u, v, w) are the cartesian components of velocity, ρ is static density, p is static pressure, c is the static speed of sound, and μ is the molecular viscosity. In this example, tilde quantities $(\tilde{\ })$ are nondimensional and all others are dimensional. Four dimensional scales are used for normalization: L (a unit length), ρ_∞, c_∞ and μ_∞. However, only three fundamental dimensional units are represented: mass, length and time. The extra normalizing scale leads to inconsistent normalization. The primary consequence of this is additional nondimensional parameters, such as Reynolds number, appearing in the nondimensionalized governing equations where none are found in the original dimensional equations. Many definitions, including skin friction coefficient, also have extra terms appearing in the nondimensionalized form. This adds unnecessary complication to any data or equation manipulation associated with the flow solver.

Consistent normalization avoids many of these problems. Here the number of scales used for normalization is the same as the number fundamental dimensional units represented by the data. Using consistent normalization, the resulting nondimensionalized form of equations and definitions is identical to their original dimensional formulations. One piece of evidence to support this assertion is that it is not possible to form any nondimensional parameters from the set of dimensional scales used for normalization.

An important fallout of consistent normalization is that the actual scales used for normalization become immaterial for all data manipulation processes. To illustrate this consider the following nondimensionalization procedure: Let M (mass), L (length) and T (time) be arbitrary dimensional scales by which all data is normalized (neglect temperature data for the present). The nondimensional data follows:

$$\begin{array}{lll} x' = x/L, & u' = u/(L/T), & \rho' = \rho/(M/L^3), \\ y' = y/L, & v' = v/(L/T), & p' = p/(M/(LT^2)), \\ z' = z/L, & w' = w/(L/T), & \mu' = \mu/(M/(LT)), \end{array}$$

where primed quantities are nondimensional and all others are dimensional.

Consider an existing object base where all field data and all reference data is nondimensional and normalized as shown. Assume the object base has a single reference state given by,

$$\begin{array}{lll} x'_{\mathrm{ref}} = x_{\mathrm{ref}}/L, & u'_{\mathrm{ref}} = u_{\mathrm{ref}}/(L/T), & \rho'_{\mathrm{ref}} = \rho_{\mathrm{ref}}/(M/L^3), \\ y'_{\mathrm{ref}} = y_{\mathrm{ref}}/L, & v'_{\mathrm{ref}} = v_{\mathrm{ref}}/(L/T), & p'_{\mathrm{ref}} = p_{\mathrm{ref}}/(M/(LT^2)) \\ z'_{\mathrm{ref}} = z_{\mathrm{ref}}/L, & w'_{\mathrm{ref}} = w_{\mathrm{ref}}/(L/T), & \mu'_{\mathrm{ref}} = \mu_{\mathrm{ref}}/(M/(LT)). \end{array}$$

If a user wanted to change the nondimensionalization of grid-point pressures, the procedure is straightforward. Let the desired new normalization be given by $p''_{ijk} = p_{ijk}/(\rho_{\mathrm{ref}} c^2_{\mathrm{ref}})$, where all terms on the right-hand-side are *dimensional*, and as such they are unknown to the object base user. However, the desired manipulation is possible using only nondimensional data provided in the object base,

$$\begin{aligned} p''_{ijk} &\equiv p_{ijk}/(\rho_{\mathrm{ref}} c^2_{\mathrm{ref}}) \\ &= \frac{p_{ijk}}{M/(LT^2)} \frac{M/L^3}{\rho_{\mathrm{ref}}} \left[\frac{L/T}{c_{\mathrm{ref}}}\right]^2 \\ &= p'_{ijk}/(\rho'_{\mathrm{ref}}(c'_{\mathrm{ref}})^2) \end{aligned}$$

Thus, the desired renormalization is possible using the object base's nondimensional data as if it were actually dimensional. There is, in fact, a high degree of equivalence between dimensional data and consistently normalized nondimensional data. The procedure shown in this example should extend to any desired renormalization, provided the needed reference-state quantities are included in the object base.

This example points out two requirements for data in the class **normalized_by_unknown_dimensional**,

a) All nondimensional data within a given object base that is of class **normalized_by_unknown_dimensional** shall be consistently normalized.

b) Any nondimensional reference state appearing in an object base should be sufficiently populated with reference quantities to allow for renormalization procedures.

These two stipulations lead to the following:

— The dimensional scales used to nondimensionalize all data are immaterial, and there is no need to identify these quantities in an object base.

— The dimensional scales need not be reference-state quantities provided in the object base.

EXAMPLE 4 A CFD analysis could contain freestream reference state conditions, but all the data is normalized by sonic conditions (which are not provided).

— All renormalization procedures can be carried out treating the data as if it were dimensional with a consistent set of units.

— Any application code that internally uses consistent normalization can use the data provided in an object base without modification or transformation to the code's internal normalization.

EXAMPLE 5 A CFD application code that internally uses inconsistent normalization could easily read and write data to a nondimensional object base that conforms to the above stipulations. On output, the code could renormalize data so it is consistently normalized. Probably, the easiest method would be to remove the molecular viscosity scale (μ_∞), and only use L, ρ_∞ and c_∞ for all normalizations (recall these are dimensional scales). The only change from the above example would be the nondimensionalization of viscosity, which would become, $\tilde{\tilde{\mu}} = \mu/(\rho_\infty c_\infty L)$. The code could then output all field data as,

$$\begin{array}{rlrlrl} \tilde{x}_{ijk} &= x_{ijk}/L, & \tilde{u}_{ijk} &= u_{ijk}/c_\infty, & \tilde{\rho}_{ijk} &= \rho_{ijk}/\rho_\infty, \\ \tilde{y}_{ijk} &= y_{ijk}/L, & \tilde{v}_{ijk} &= v_{ijk}/c_\infty, & \tilde{p}_{ijk} &= p_{ijk}/(\rho_\infty c_\infty^2), \\ \tilde{z}_{ijk} &= z_{ijk}/L, & \tilde{w}_{ijk} &= w_{ijk}/c_\infty, & \tilde{\tilde{\mu}}_{ijk} &= \mu_{ijk}/(\rho_\infty c_\infty L), \end{array}$$

and output the freestream reference quantities,

$$\begin{array}{rlrl} \tilde{u}_\infty &= u_\infty/c_\infty, & \tilde{\rho}_\infty &= \rho_\infty/\rho_\infty = 1, \\ \tilde{v}_\infty &= v_\infty/c_\infty, & \tilde{p}_\infty &= p_\infty/(\rho_\infty c_\infty^2) = 1/\gamma, \\ \tilde{w}_\infty &= w_\infty/c_\infty, & \tilde{\tilde{\mu}}_\infty &= \mu_\infty/(\rho_\infty c_\infty L) \sim O(1/Re), \end{array}$$

where γ is the specific heat ratio (assumes a perfect gas) and Re is the Reynolds number.

On input, the flow solver should be able to recover its internal normalizations from the data in a nondimensional object base by treating the data as if it were dimensional.

— **nondimensional_parameter:** When the data class is **nondimensional_parameter**, the data is a nondimensional parameter (or array of nondimensional parameters). Examples include Mach number, Reynolds number and pressure coefficient. These parameters are prevalent in CFD, although their definitions tend to vary between different application codes.

Nondimensional parameters are distinguished from other data classes by the fact that they are *always* dimensionless. In a completely nondimensional object base, they are distinct in that their normalization is not necessarily consistent with other data.

Typically, the **units**, **exponents** and **conversion** qualifiers are not used for nondimensional parameters; although, there are a few situations where they may be used (these are discussed below). Rather than rely on optional qualifiers to describe the normalization, any nondimensional parameter shall be accompanied by their defining scales;

EXAMPLE 6 An example is Reynolds number defined as $Re = VL/\nu$, where V, L and ν are velocity, length and viscosity scales, respectively. Note that these defining scales may be dimensional or nondimensional data.

In certain situations, it may be more convenient to use the optional qualifiers of **data_array** to describe the normalization used in nondimensional parameters. These situations must satisfy two requirements: First, the defining scales are dimensional; and second, the nondimensional parameter is a normalization of a single 'raw' data quantity and it is clear what this raw data is.

EXAMPLE 7 Examples that satisfy this second constraint are pressure coefficient, where the raw data is static pressure, and lift coefficient, where the raw data is the lift force. Conversely, Reynolds number is a parameter that violates the second requirement — there are three pieces of raw data rather than one that make up *Re*.

For nondimensional parameters that satisfy these two requirements, the qualifiers **units**, **exponents** and **conversion** may be used to recover the raw dimensional data.

- **dimensionless_constant:** When the data class is **dimensionless_constant**, the data is a constant (or array of constants) with no associated dimensional units. The **units**, **exponents** and **conversion** qualifiers are not used.

EXPRESS specification:

```
*)
ENTITY data_array;
--  SUBTYPE OF (property_distribution_description);
  annotation   : info;
  dimension    : INTEGER;
  sizes        : ARRAY [1:dimension] OF INTEGER;
  data_class   : data_class;
  data_name    : data_name;
  units        : OPTIONAL dimensional_units;
  exponents    : OPTIONAL dimensional_exponents;
  conversion   : OPTIONAL data_conversion;
  data         : ARRAY [1:size] OF REAL;
DERIVE
  size : INTEGER := multi_array_size(SELF);
END_ENTITY;
(*
```

Attribute definitions:

descriptions: is annotation;

dimension: the number of dimensions in the multidimensional data array;

sizes: the array sizes for each dimension;

data_class: the class of the data;

data_name: identifies the kind of the data;

units: the dimensional units, if any, of the data;

exponents: the dimensional exponents of the dimensional units;

conversion: the normalization of dimensional data, if any.

G.3.4.61 geometry_reference

Reference to geometry data.

EXPRESS specification:

```
*)
ENTITY geometry_reference;
  descriptions : LIST OF TEXT;
  location     : external;
END_ENTITY;
(*
```

Attribute definitions:

descriptions: is descriptive annotation;

location: is where the data can be found.

G.3.4.62 info

info is general textual information.

EXPRESS specification:

```
*)
ENTITY info;
  id          : IDENTIFIER;
  name        : LABEL;
  description : LIST OF TEXT;
END_ENTITY;
(*
```

Attribute definitions:

id: is an identifier;

name: is a user-specified instance identifier;

description: is annotation.

G.3.4.63 external

external is a reference to data that is not defined in in this model, but which is defined elsewhere.

EXPRESS specification:

```
*)
ENTITY external;
END_ENTITY;
(*
```

G.3.5 analysis_arm function definitions

G.3.5.1 multi_array_size

The **multi_array_size** function calculates the size of the **data** array in a **data_array**.

EXPRESS specification:

```
*)
FUNCTION multi_array_size(arg: data_array) : INTEGER;
```

```
LOCAL
  result : INTEGER := 1;
END_LOCAL;
  REPEAT i := 1 TO arg.dimension;
    result := result * arg.sizes[i];
  END_REPEAT;
  RETURN(result);
END_FUNCTION;
(*
```

Argument definitions:

arg: an instance of **data_array**;

RETURNS: the size of the **data** array.

EXPRESS specification:

```
*)
END_SCHEMA; -- end of analysis_arm
(*
```

Annex H
(informative)
AIM EXPRESS-G

The diagrams in this annex correspond to the AIM EXPRESS expanded listing given in annex A. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex D of ISO 10303-11.

(TBD — FIGURE RANGE)

Annex J
(informative)
AIM EXPRESS listing

This annex provides a listing of the complete EXPRESS schema specified in annex A of this part of ISO 10303 without comments or explanatory text. It also provides a listing of the EXPRESS entity names and corresponding short names as specified in annex B of this part of ISO 10303. The content of this annex is available in computer-interpretable form and can be found at the following URLs:

— Short names: <http://www.mel.nist.gov/div826/subject/apde/snr/>

— EXPRESS: <http://www.mel.nist.gov/step/parts/part237/wd/>

If there is difficulty accessing these sites contact ISO Central Secretariat or contact the ISO TC 184/SC4 Secretariat directly at: sc4sec@cme.nist.gov.

NOTE The information provided in computer-interpretable form at the above URLs is informative. The information that is contained in the body of this part of ISO 10303 is normative.

Bibliography

[1] ALLMARA, S. and McCARTHY, D., *CGNS Standard Interface Structures,* 11 August 1999

[2] POIRIER, D., *SIDS additions/modifications to support unstructured meshes and geometry links,* June 1999

[3] CGNS PROJECT GROUP, *The CFD General Notation System: Standard Interface Data Structures,* August 2000

[4] WHITE, F. M., *Viscous Fluid Flow* McGraw-Hill, 1974

[5] *IDEF0 (ICAM Definition Language 0),* Federal Information Processing Standards Publication 183, Integration Definition for Information Modeling (IDEF0), FIPS PUB 183, National Institute for Standards and Technology, December 1993.

Index

additional_design_information (entity) . . . 115
advanced_boundary_representation (entity) . . . 111
alternate_part (entity) . . . 104
analysis (entity) . . . 108
analysis_arm (schema) . . . 117
analysis_design_version_relationship (entity) . . . 106
analysis_discipline_product_definition (entity) . . . 107
analysis_model (entity) . . . 166
analysis_shape (entity) . . . 114
analysis_step (entity) . . . 167
analysis_version (entity) . . . 106
approval (entity) . . . 96
assembly (entity) . . . 91
bc_data (entity) . . . 189
bc_dataset (entity) . . . 188
bc_type (select) . . . 147
bc_type_compound (enumeration) . . . 150
bc_type_simple (enumeration) . . . 148
behaviour_model (entity) . . . 193
boundary_condition (entity) . . . 182
cfd_data_name (select) . . . 141
change_order (entity) . . . 88
change_request (entity) . . . 90
component_assembly_position (entity) . . . 94
condition (entity) . . . 181
coordinate_data_name (enumeration) . . . 145
data_array (entity) . . . 205
data_class (select) . . . 144
data_conversion (entity) . . . 200
data_name (select) . . . 140
DATE (entity) . . . 85
date_effectivity (entity) . . . 100
defined_data_class (select) . . . 144
defined_data_name (select) . . . 140
design_discipline_product_definition (entity) . . . 107
design_or_analysis (select) . . . 84
design_specification (entity) . . . 116
diffusion_equation (entity) . . . 193
diffusion_model (entity) . . . 194
dimensional_exponents (entity) . . . 201
dimensional_units (entity) . . . 200
discrete_data_with_rind (entity) . . . 199
effectivity (entity) . . . 97
element_list_bc (entity) . . . 184
element_range_bc (entity) . . . 184
elements_bc (entity) . . . 184
equation (entity) . . . 191
equation_set (entity) . . . 192
external (entity) . . . 206
faceted_boundary_representation (entity) . . . 111
family (entity) . . . 191

FILE (entity) 86
flow_solution_data_name (enumeration) 164
force_moment_data_name (enumeration) 153
gas_model (entity) 195
gas_model_data_name (enumeration) 155
gas_model_type (enumeration) 154
geometric_model_representation (entity) 110
geometry_element (entity) 108
geometry_reference (entity) 205
governing_equation (entity) 193
governing_equation_type (enumeration) 153
grid_coordinates (entity) 174
idealized_analysis_shape (entity) 114
IDENTIFIER (entity) 85
ijk_minmax (enumeration) 150
indices_group (type) 135
indices_list (entity) 173
indices_range (entity) 173
info (entity) 206
integral_data (entity) 191
LABEL (entity) 86
lot_effectivity (entity) 100
make_from (entity) 95
management_arm (schema) 83
manifold_surface_with_topology (entity) 111
MASS_MEASURE (entity) 86
matched_mesh_connection (entity) 176
mesh (entity) 170
mesh_abutting (entity) 178
mesh_connectivity (entity) 175
mesh_location (type) 135
mesh_overset (entity) 179
mesh_overset_hole (entity) 179
mismatched_donor_mesh (entity) 179
mismatched_mesh_connection (entity) 177
mismatched_mesh_region (entity) 178
multi_array_size (function) 206
multiple_mesh_block (entity) 174
next_higher_assembly (entity) 93
node_shape (entity) 115
nominal_design_shape (entity) 113
non_topological_surface_and_wireframe (entity) 112
nondimensional_parameter_name (enumeration) 142
part (entity) 104
part_version (entity) 105
person_organization (entity) 97
point_list_bc (entity) 185
point_model (entity) 112
point_range_bc (entity) 185
process_specification (entity) 117
product_configuration (entity) 101
product_model (entity) 102

promissory_usage (entity) 93
reference_state (entity) 190
result (entity) 197
retention_data_select (select) 85
retention_period (entity) 101
Riemann_1D_data_name (enumeration) 146
rind (entity) 171
sc1_analysis_shape (subtype_constraint) 114
sc1_assembly (subtype_constraint) 91
sc1_bc_dataset (subtype_constraint) 188
sc1_behaviour_model (subtype_constraint) 193
sc1_boundary_condition (subtype_constraint) 182
sc1_condition (subtype_constraint) 181
sc1_effectivity (subtype_constraint) 97
sc1_equation (subtype_constraint) 191
sc1_geometric_model_representation (subtype_constraint) 110
sc1_mesh (subtype_constraint) 170
sc1_mesh_connectivity (subtype_constraint) 175
sc1_mismatched_donor_mesh (subtype_constraint) 179
sc1_mismatched_mesh_connection (subtype_constraint) 177
sc1_mismatched_mesh_region (subtype_constraint) 178
sc1_result (subtype_constraint) 197
sc1_shape (subtype_constraint) 113
sc1_specification (subtype_constraint) 116
sc1_structured_mesh (subtype_constraint) 171
sc1_unstructured_mesh (subtype_constraint) 172
sc1_work_order (subtype_constraint) 88
sc1_work_request (subtype_constraint) 89
sc1_zone (subtype_constraint) 168
sequence_effectivity (entity) 98
shape (entity) 113
shape_aspect (entity) 112
solution (entity) 197
solution_with_rind (entity) 198
specification (entity) 116
start_order (entity) 89
start_request (entity) 90
structured_donor_mesh (entity) 180
structured_mesh (entity) 171
structured_zone (entity) 169
substitute_part (entity) 94
supplied_part_version (entity) 94
supplier (entity) 95
surface_finish_specification (entity) 116
TEXT (entity) 86
thermal_conductivity_model (entity) 195
thermal_conductivity_model_data_name (enumeration) 158
thermal_conductivity_model_type (enumeration) 157
turbulence_closure (entity) 196
turbulence_closure_data_name (enumeration) 159
turbulence_closure_type (enumeration) 159
turbulence_model (entity) 196

turbulence_model_data_name (enumeration) 161
turbulence_model_type (enumeration) 160
unstructured_donor_mesh (entity) 180
unstructured_mesh (entity) 172
unstructured_zone (entity) 169
vertex_defined_cell (entity) 172
viscosity_model (entity) 196
viscosity_model_data_name (enumeration) 156
viscosity_model_type (enumeration) 155
wireframe_with_topology (entity) 112
work_order (entity) 88
work_request (entity) 89
zone (entity) 168
zone_bc (entity) 181