INTERNATIONAL STANDARD

ISO 10303-41

First edition
1994-12-15

Industrial automation systems and integration — Product data representation and exchange —

Part 41:
Integrated generic resources: Fundamentals of product description and support

Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits —

Partie 41: Ressources génériques intégrées: Principes de description et de support de produits

Reference number
ISO 10303-41:1994(E)

Contents

Page

Section 1 : General 1

1.1 Scope 1
1.1.1 Generic product description resources 1
1.1.2 Generic management resources 2
1.1.3 Support resources 2

1.2 Normative references 2

1.3 Definitions and abbreviations 3
1.3.1 Terms defined in ISO 10303-1 3
1.3.2 Terms defined in ISO 8601 3
1.3.3 Abbreviations defined in ISO 1000 4

Section 2 : Generic product description resources 5

2.1 Introduction 5

2.2 application_context_schema 5
2.2.1 Introduction 6
2.2.2 Fundamental concepts and assumptions 6
2.2.3 application_context_schema entity definitions 6
2.2.3.1 application_context 6
2.2.3.2 application_protocol_definition 7
2.2.3.3 application_context_element 8
2.2.3.4 product_context 8
2.2.3.5 product_definition_context 9
2.2.3.6 product_concept_context 9
2.2.3.7 library_context 10

2.3 product_definition_schema 11
2.3.1 Introduction 11
2.3.2 Fundamental concepts and assumptions 12

International Organization for Standardization
Case Postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

2.3.3 product_definition_schema type definition: source . . . 12
2.3.4 product_definition_schema entity definitions . . . 13
2.3.4.1 product . . . 13
2.3.4.2 product_category . . . 14
2.3.4.3 product_related_product_category . . . 14
2.3.4.4 product_category_relationship . . . 14
2.3.4.5 product_definition_formation . . . 16
2.3.4.6 product_definition_formation_relationship . . . 16
2.3.4.7 product_definition_formation_with_specified_source . . . 17
2.3.4.8 product_definition . . . 18
2.3.4.9 product_definition_with_associated_documents . . . 18
2.3.4.10 product_definition_relationship . . . 19
2.3.4.11 product_definition_substitute . . . 20
2.3.4.12 product_definition_effectivity . . . 21
2.3.5 product_definition_schema function definitions . . . 21
2.3.5.1 acyclic_product_definition_formation_relationship . . . 21
2.3.5.2 acyclic_product_definition_relationship . . . 23
2.3.5.3 acyclic_product_category_relationship . . . 24

2.4 product_property_definition_schema . . . 25
2.4.1 Introduction . . . 26
2.4.2 Fundamental concepts and assumptions . . . 26
2.4.3 product_property_definition_schema type definitions . . . 26
2.4.3.1 characterized_definition . . . 26
2.4.3.2 characterized_product_definition . . . 27
2.4.3.3 shape_definition . . . 27
2.4.4 product_property_definition_schema entity definitions . . . 28
2.4.4.1 characterized_object . . . 28
2.4.4.2 property_definition . . . 28
2.4.4.3 product_definition_shape . . . 29
2.4.4.4 shape_aspect . . . 30
2.4.4.5 shape_aspect_relationship . . . 30
2.4.5 product_property_definition_schema function definition: acyclic_shape_aspect_relationship . . . 32

2.5 product_property_representation_schema . . . 33
2.5.1 Introduction . . . 34
2.5.2 Fundamental concepts and assumptions . . . 34
2.5.3 product_property_representation_schema entity definitions . . . 35
2.5.3.1 shape_representation . . . 35
2.5.3.2 property_definition_representation . . . 35
2.5.3.3 shape_representation_relationship . . . 35
2.5.3.4 context_dependent_shape_representation . . . 36
2.5.3.5 shape_definition_representation . . . 37
2.5.4 product_property_representation_schema function definitions . . . 37

2.5.4.1 relatives_of_product_definitions . . . 37
2.5.4.2 relatives_of_shape_representations . . . 39

Section 3 : Management resources . . . 41

3.1 Introduction . . . 41

3.2 management_resources_schema . . . 41
3.2.1 Introduction . . . 42
3.2.2 Fundamental concepts and assumptions . . . 42
3.2.3 management_resources_schema entity definitions . . . 42
3.2.3.1 name_assignment . . . 42
3.2.3.2 external_referent_assignment . . . 43
3.2.3.3 library_assignment . . . 43
3.2.3.4 document_reference . . . 44
3.2.3.5 action_request_assignment . . . 44
3.2.3.6 action_assignment . . . 45
3.2.3.7 certification_assignment . . . 45
3.2.3.8 approval_assignment . . . 45
3.2.3.9 contract_assignment . . . 46
3.2.3.10 security_classification_assignment . . . 46
3.2.3.11 person_assignment . . . 46
3.2.3.12 organization_assignment . . . 47
3.2.3.13 person_and_organization_assignment . . . 47
3.2.3.14 date_assignment . . . 48
3.2.3.15 time_assignment . . . 48
3.2.3.16 date_and_time_assignment . . . 49
3.2.3.17 group_assignment . . . 49
3.2.3.18 effectivity_assignment . . . 49

Section 4 : Support resources . . . 51

4.1 Introduction . . . 51

4.2 document_schema . . . 52
4.2.1 Introduction . . . 52
4.2.2 Fundamental concepts and assumptions . . . 53
4.2.3 document_schema entity definitions . . . 53
4.2.3.1 document_type . . . 53
4.2.3.2 document . . . 53
4.2.3.3 document_with_class . . . 54
4.2.3.4 document_usage_constraint . . . 54
4.2.3.5 document_relationship . . . 55
4.2.4 document_schema function definition: acyclic_document_relationship . . . 56

4.3 action_schema . . . 57

4.3.1 Introduction . . . 57
4.3.2 Fundamental concepts and assumptions . . . 58
4.3.3 action_schema type definition: supported_item . . . 58
4.3.4 action_schema entity definitions . . . 58
4.3.4.1 action . . . 58
4.3.4.2 executed_action . . . 59
4.3.4.3 directed_action . . . 59
4.3.4.4 action_status . . . 59
4.3.4.5 action_request_status . . . 59
4.3.4.6 action_relationship . . . 60
4.3.4.7 action_method . . . 60
4.3.4.8 action_request_solution . . . 61
4.3.4.9 action_method_relationship . . . 61
4.3.4.10 versioned_action_request . . . 62
4.3.4.11 action_directive . . . 62
4.3.4.12 action_resource . . . 63
4.3.4.13 action_resource_relationship . . . 63
4.3.4.14 action_resource_type . . . 64
4.3.5 action_schema function definitions . . . 64
4.3.5.1 acyclic_action_relationship . . . 64
4.3.5.2 acyclic_action_resource_relationship . . . 65
4.3.5.3 acyclic_action_method_relationship . . . 67

4.4 certification_schema . . . 68
4.4.1 Introduction . . . 69
4.4.2 Fundamental concepts and assumptions . . . 69
4.4.3 certification_schema entity definitions . . . 69
4.4.3.1 certification_type . . . 69
4.4.3.2 certification . . . 69

4.5 approval_schema . . . 70
4.5.1 Introduction . . . 71
4.5.2 Fundamental concepts and assumptions . . . 71
4.5.3 approval_schema entity definitions . . . 71
4.5.3.1 approval_status . . . 71
4.5.3.2 approval . . . 71
4.5.3.3 approval_date_time . . . 72
4.5.3.4 approval_person_organization . . . 72
4.5.3.5 approval_role . . . 73
4.5.3.6 approval_relationship . . . 73
4.5.4 approval_schema function definition: acyclic_approval_relationship . . . 73

4.6 contract_schema . . . 75
4.6.1 Introduction . . . 75
4.6.2 Fundamental concepts and assumptions . . . 75

4.6.3 contract_schema entity definitions . . . 76
4.6.3.1 contract_type . . . 76
4.6.3.2 contract . . . 76

4.7 security_classification_schema . . . 76
4.7.1 Introduction . . . 77
4.7.2 Fundamental concepts and assumptions . . . 77
4.7.3 security_classification_schema entity definitions . . . 77
4.7.3.1 security_classification_level . . . 77
4.7.3.2 security_classification . . . 78

4.8 person_organization_schema . . . 78
4.8.1 Introduction . . . 79
4.8.2 Fundamental concepts and assumptions . . . 79
4.8.3 person_organization_schema type definition: person_organization_select . . . 79
4.8.4 person_organization_schema entity definitions . . . 80
4.8.4.1 address . . . 80
4.8.4.2 personal_address . . . 81
4.8.4.3 organizational_address . . . 81
4.8.4.4 person . . . 82
4.8.4.5 organization . . . 82
4.8.4.6 organizational_project . . . 83
4.8.4.7 person_and_organization . . . 83
4.8.4.8 organization_relationship . . . 84
4.8.4.9 person_and_organization_role . . . 84
4.8.4.10 person_role . . . 85
4.8.4.11 organization_role . . . 85
4.8.5 person_organization_schema function definition: acyclic_organization_relationship . . . 85

4.9 date_time_schema . . . 87
4.9.1 Introduction . . . 87
4.9.2 Fundamental concepts and assumptions . . . 87
4.9.3 date_time_schema type definitions . . . 87
4.9.3.1 date_time_select . . . 87
4.9.3.2 year_number . . . 88
4.9.3.3 month_in_year_number . . . 88
4.9.3.4 week_in_year_number . . . 88
4.9.3.5 day_in_week_number . . . 89
4.9.3.6 day_in_month_number . . . 89
4.9.3.7 day_in_year_number . . . 90
4.9.3.8 ahead_or_behind . . . 90
4.9.3.9 hour_in_day . . . 90
4.9.3.10 minute_in_hour . . . 91
4.9.3.11 second_in_minute . . . 91
4.9.4 date_time_schema entity definitions . . . 91

4.9.4.1 date . . . 91
4.9.4.2 calendar_date . . . 92
4.9.4.3 ordinal_date . . . 92
4.9.4.4 week_of_year_and_day_date . . . 93
4.9.4.5 coordinated_universal_time_offset . . . 93
4.9.4.6 local_time . . . 94
4.9.4.7 date_and_time . . . 94
4.9.4.8 date_time_role . . . 95
4.9.4.9 date_role . . . 95
4.9.4.10 time_role . . . 95
4.9.5 date_time_schema function definitions . . . 96
4.9.5.1 leap_year . . . 96
4.9.5.2 valid_calendar_date . . . 96
4.9.5.3 valid_time . . . 97

4.10 group_schema . . . 99
4.10.1 Introduction . . . 99
4.10.2 Fundamental concepts and assumptions . . . 99
4.10.3 group_schema entity definitions . . . 100
4.10.3.1 group . . . 100
4.10.3.2 group_relationship . . . 100
4.10.4 group_schema function definition:
acyclic_group_relationship . . . 101

4.11 effectivity_schema . . . 102
4.11.1 Introduction . . . 103
4.11.2 Fundamental concepts and assumptions . . . 103
4.11.3 effectivity_schema entity definitions . . . 103
4.11.3.1 effectivity . . . 103
4.11.3.2 serial_numbered_effectivity . . . 104
4.11.3.3 dated_effectivity . . . 104
4.11.3.4 lot_effectivity . . . 105

4.12 external_reference_schema . . . 105
4.12.1 Introduction . . . 106
4.12.2 Fundamental concepts and assumptions . . . 106
4.12.3 external_reference_schema type definitions . . . 106
4.12.3.1 message . . . 106
4.12.3.2 reference . . . 107
4.12.4 external_reference_schema entity definitions . . . 107
4.12.4.1 external_source . . . 107
4.12.4.2 external_source_relationship . . . 107
4.12.4.3 pre_defined_item . . . 108
4.12.4.4 externally_defined_item . . . 108
4.12.5 external_reference_schema function definition:
acyclic_external_source_relationship . . . 109
4.12.6 End of schema declaration . . . 110

4.13 support_resource_schema 110
4.13.1 Introduction 111
4.13.2 Fundamental concepts and assumptions 111
4.13.3 support_resource_schema type definitions 111
4.13.3.1 identifier 111
4.13.3.2 label 111
4.13.3.3 text 112
4.13.4 support_resource_schema function definition:
bag_to_set 112

4.14 measure_schema 113
4.14.1 Introduction 113
4.14.2 Fundamental concepts and assumptions 114
4.14.3 measure_schema type definitions 114
4.14.3.1 measure_value 114
4.14.3.2 length_measure 115
4.14.3.3 mass_measure 115
4.14.3.4 time_measure 115
4.14.3.5 electric_current_measure 115
4.14.3.6 thermodynamic_temperature_measure 116
4.14.3.7 amount_of_substance_measure 116
4.14.3.8 luminous_intensity_measure 116
4.14.3.9 plane_angle_measure 116
4.14.3.10 solid_angle_measure 116
4.14.3.11 area_measure 117
4.14.3.12 volume_measure 117
4.14.3.13 ratio_measure 117
4.14.3.14 parameter_value 117
4.14.3.15 numeric_measure 118
4.14.3.16 positive_length_measure 118
4.14.3.17 positive_plane_angle_measure 118
4.14.3.18 positive_ratio_measure 118
4.14.3.19 context_dependent_measure 119
4.14.3.20 descriptive_measure 119
4.14.3.21 count_measure 119
4.14.3.22 unit 119
4.14.3.23 si_unit_name 120
4.14.3.24 si_prefix 122
4.14.4 measure_schema entity definitions 123
4.14.4.1 named_unit 123
4.14.4.2 si_unit 123
4.14.4.3 conversion_based_unit 124
4.14.4.4 context_dependent_unit 124
4.14.4.5 length_unit 125
4.14.4.6 mass_unit 125
4.14.4.7 time_unit 126

4.14.4.8 electric_current_unit . . . 126
4.14.4.9 thermodynamic_temperature_unit . . . 127
4.14.4.10 amount_of_substance_unit . . . 127
4.14.4.11 luminous_intensity_unit . . . 128
4.14.4.12 plane_angle_unit . . . 128
4.14.4.13 solid_angle_unit . . . 129
4.14.4.14 area_unit . . . 129
4.14.4.15 volume_unit . . . 130
4.14.4.16 ratio_unit . . . 130
4.14.4.17 dimensional_exponents . . . 131
4.14.4.18 derived_unit_element . . . 132
4.14.4.19 derived_unit . . . 132
4.14.4.20 global_unit_assigned_context . . . 133
4.14.4.21 measure_with_unit . . . 133
4.14.4.22 length_measure_with_unit . . . 134
4.14.4.23 mass_measure_with_unit . . . 134
4.14.4.24 time_measure_with_unit . . . 135
4.14.4.25 electric_current_measure_with_unit . . . 135
4.14.4.26 thermodynamic_temperature_measure_with_unit . . . 135
4.14.4.27 amount_of_substance_measure_with_unit . . . 136
4.14.4.28 luminous_intensity_measure_with_unit . . . 136
4.14.4.29 plane_angle_measure_with_unit . . . 137
4.14.4.30 solid_angle_measure_with_unit . . . 137
4.14.4.31 area_measure_with_unit . . . 137
4.14.4.32 volume_measure_with_unit . . . 138
4.14.4.33 ratio_measure_with_unit . . . 138
4.14.5 measure_schema function definitions . . . 139
4.14.5.1 dimensions_for_si_unit . . . 139
4.14.5.2 derive_dimensional_exponents . . . 140
4.14.5.3 valid_units . . . 141

Annexes

A Short names of entities . . . 145

B Information object registration . . . 151
B.1 Document identification . . . 151
B.2 Schema identification . . . 151
B.2.1 application_context_schema identification . . . 151
B.2.2 product_definition_schema identification . . . 151
B.2.3 product_property_definition_schema identification . . . 151
B.2.4 product_property_representation_schema identification . . . 151
B.2.5 management_resources_schema identification . . . 152
B.2.6 document_schema identification . . . 152
B.2.7 action_schema identification . . . 152
B.2.8 certification_schema identification . . . 152
B.2.9 approval_schema identification . . . 152

B.2.10 contract_schema identification . . . 152
B.2.11 security_classification_schema identification . . . 152
B.2.12 person_organization_schema identification . . . 152
B.2.13 date_time_schema identification . . . 153
B.2.14 group_schema identification . . . 153
B.2.15 effectivity_schema identification . . . 153
B.2.16 external_reference_schema identification . . . 153
B.2.17 support_resource_schema identification . . . 153
B.2.18 measure_schema identification . . . 153

C Computer-interpretable listings . . . 154

D Technical discussions . . . 155
D.1 Generic product description resource structure . . . 155
D.2 Acyclicity avoidance function template . . . 155
D.2.1 acyclic_object_relationship . . . 155
D.3 Relationship template . . . 157
D.3.1 object_relationship . . . 157

E Examples . . . 159
E.1 Use of the **product_definition_schema** . . . 159
E.2 Use of the generic management resource constructs . . . 159

F EXPRESS-G diagrams . . . 161

G Bibliography . . . 177

Figures

1 The groupings of resource schemas into generic product description resources, generic management resources, and support resources . . . xv
D.1 The structure of the generic product description resource . . . 156
F.1 application_context_schema - EXPRESS-G diagram 1 of 1 . . . 162
F.2 product_definition_schema - EXPRESS-G diagram 1 of 1 . . . 163
F.3 product_property_definition_schema - EXPRESS-G diagram 1 of 1 . . . 164
F.4 product_property_representation_schema - EXPRESS-G diagram 1 of 1 . . . 165
F.5 management_resources_schema - EXPRESS-G diagram 1 of 1 . . . 166
F.6 document_schema - EXPRESS-G diagram 1 of 1 . . . 167
F.7 action_schema - EXPRESS-G diagram 1 of 1 . . . 168
F.8 certification_schema - EXPRESS-G diagram 1 of 1 . . . 168
F.9 approval_schema - EXPRESS-G diagram 1 of 1 . . . 169
F.10 contract_schema - EXPRESS-G diagram 1 of 1 . . . 169

F.11 security_classification schema - EXPRESS-G diagram 1 of 1 170
F.12 person_organization_schema - EXPRESS-G diagram 1 of 1 170
F.13 date_time_schema - EXPRESS-G diagram 1 of 1 171
F.14 group_schema - EXPRESS-G diagram 1 of 1 172
F.15 effectivity_schema - EXPRESS-G diagram 1 of 1 173
F.16 external_reference_schema - EXPRESS-G diagram 1 of 1 174
F.17 support_resource_schema - EXPRESS-G diagram 1 of 1 175
F.18 measure_schema - EXPRESS-G diagram 1 of 3 175
F.19 measure_schema - EXPRESS-G diagram 2 of 3 176
F.20 measure_schema - EXPRESS-G diagram 3 of 3 176

Tables

A.1 Short names of entities 145

Foreword

The International Organization for Standardization (ISO) 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.

Draft International Standards adopted by 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.

International Standard ISO 10303–41 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration,* Subcommittee SC4, *Industrial data and global manufacturing programming languages.*

ISO 10303 consists of the following parts under the general title *Industrial automation systems and integration – Product data representation and exchange:*

- Part 1, Overview and fundamental principles;
- Part 11, Description methods: The *EXPRESS* language reference manual;
- Part 21, Implementation methods: Clear text encoding of the exchange structure;
- Part 22, Implementation methods: Standard data access interface specification;
- Part 31, Conformance testing methodology and framework: General concepts;
- Part 32, Conformance testing methodology and framework: Requirements on testing laboratories and clients;
- Part 41, Integrated generic resources: Fundamentals of product description and support;
- Part 42, Integrated generic resources: Geometric and topological representation;
- Part 43, Integrated generic resources: Representation structures;
- Part 44, Integrated generic resources: Product structure configuration;
- Part 45, Integrated generic resources: Materials;
- Part 46, Integrated generic resources: Visual presentation;
- Part 47, Integrated generic resources: Shape variation tolerances;
- Part 49, Integrated generic resources: Process structure and properties;

- Part 101, Integrated application resources: Draughting;
- Part 104, Integrated application resources: Finite element analysis;
- Part 105, Integrated application resources: Kinematics;
- Part 201, Application protocol: Explicit draughting;
- Part 202, Application protocol: Associative draughting;
- Part 203, Application protocol: Configuration controlled design;
- Part 207, Application protocol: Sheet metal die planning and design;
- Part 210, Application protocol: Printed circuit assembly product design data;
- Part 213, Application protocol: Numerical control process plans for machined parts.

The structure of this International Standard is described in ISO 10303-1. The numbering of the parts of this International Standard reflects its structure:

- Part 11 specifies the description methods;
- Parts 21 and 22 specify the implementation methods;
- Parts 31 and 32 specify the conformance testing methodology and framework;
- Parts 41 to 49 specify the integrated generic resources;
- Parts 101 to 105 specify the integrated application resources;
- Parts 201 to 213 specify the application protocols.

Should further parts be published, they will follow the same numbering pattern.

Annexes A and B form an integral part of this part of ISO 10303. Annexes C, D, E and F are for information only.

Diskette

Users should note that this part of ISO 10303 comprises a diskette:
— the short names of entities given in annex A are also included on the diskette;
— the EXPRESS listings (annex C) are provided on the diskette only;
— a method to enable users to report errors in the documentation is given. Full details are provided in the file.

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.

This International Standard is organized as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series: description methods, integrated resources, application protocols, abstract test suites, implementation methods, and conformance testing. The series are described in ISO 10303–1. This part of ISO 10303 is a member of the integrated resources series. Major subdivisions of this International Standard are:

- generic product description resources;
- generic management resources;
- support resources.

The groupings of resource schemas into these major subdivisions are shown in figure 1.

The generic product description resources provide an overall organization for the integrated resources that are documented in other parts of ISO 10303. They support the description of application-independent facts that are common to all products. In this part the combination of the generic product description resources and the ISO 10303 integrated resources that are defined in the other parts that belong to the integrated resources class is referred to as the "integrated product description resources".

The generic management resources support the description of information that is used to manage and control product data. Together, the integrated product description resources and the generic management resources are the foundations upon which application interpreted models, the normative conceptual schemas of application protocols, are built. Application interpreted models apply selected generic management resources to elements of the integrated product description resources to satisfy the requirements that are specified in the appropriate application reference model.

The support resources are a set of shared resource constructs that are used by the ISO 10303 integrated resources. They provide an underlying consistency across the resources of ISO 10303.

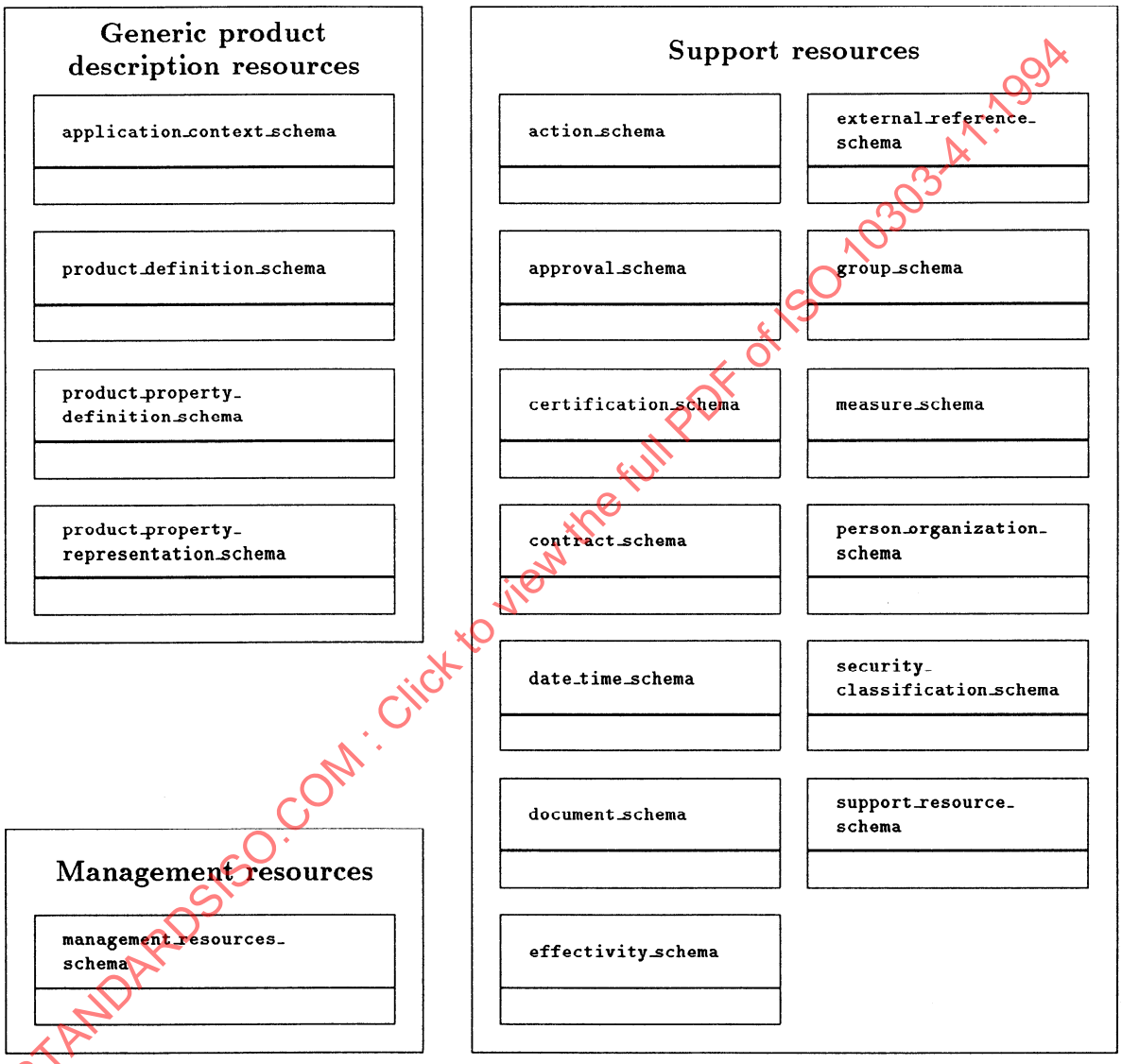

Figure 1 – The groupings of resource schemas into generic product description resources, generic management resources, and support resources

This page intentionally left blank

Industrial automation systems and integration — Product data representation and exchange — Part 41 : Integrated generic resources: Fundamentals of product description and support

Section 1 : General

1.1 Scope

This part of ISO 10303 specifies the following:

- generic product description resources (section 2);
- generic management resources (section 3);
- support resources (section 4).

The schemas which are specified in this part of ISO 10303 are organized according to these sections.

1.1.1 Generic product description resources

This section of ISO 10303-41 specifies the resource constructs for the high level structure for the representation of products and their properties. It also specifies ISO 10303 integrated resources for the description of generic aspects of product usage, categorization of products and associations between products.

The following are within scope in this section:

- the identification of a product;
- the categorization of products;
- the specification of definitions of, relationships among, and allowable substitutions for a product;
- the specification of the representation of the shape of a product;
- the specification of the representation of the properties of a product;
- the description of the application context for which product data is defined.

1.1.2 Generic management resources

This section of ISO 10303-41 specifies the resource constructs for the structures that are used to associate administrative data with product data.

The following are within scope in this section:

- the structure for connecting product data within an application context to related administrative data.

1.1.3 Support resources

This section of ISO 10303-41 specifies the resource constructs for administrative data, physical quantities and their units, and basic data types.

The following are within scope in this section:

- the description of references to documents;
- the descriptions of actions, action requests, and the status of actions;
- the descriptions of certification, approvals, security classifications, and effectivities;
- the identification of contracts;
- the identification of people and organizations;
- the specification of dates and times;
- the provision of mechanisms for grouping items and referring to information that is defined outside an exchange;
- the definition of physical quantities and their units.

1.2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303.. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO 10303. are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 31-0:1992, *Quantities and units.*

ISO 1000:1981, *SI units and recommendations for the use of their multiples and of certain other units.*

ISO 8601:1988, *Date elements and interchange formats – Information interchange – Representations of dates and times.*

ISO 8824-1:–[1)], *Information Technology – Open Systems Interconnection – Abstract Syntax Notation One (ASN.1) – Part 1: Specification of Basic Notation.*

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

ISO 10303-11:1994, *Industrial automation systems and integration – Product data representation and exchange – Part 11: EXPRESS language reference manual.*

ISO 10303-43:1994, *Industrial automation systems and integration – Product data representation and exchange – Part 43: Integrated resources: Representation structures.*

ISO 10303-44:1994, *Industrial automation systems and integration – Product data representation and exchange – Part 44: Integrated resources: Product structure configuration.*

1.3 Definitions and abbreviations

1.3.1 Terms defined in ISO 10303-1

This Part of ISO 10303 makes use of the following terms defined in ISO 10303-1:

- application;
- application context;
- application interpreted model;
- application protocol;
- application reference model;
- interpretation;
- product.

1.3.2 Terms defined in ISO 8601

This part of ISO 10303 makes use of the following terms defined in ISO 8601:

- calendar date;

[1)] To be published.

- ordinal date;
- second;
- minute;
- hour;
- day;
- week;
- calendar week;
- month;
- year;
- calendar year;
- common year;
- leap year;
- local time;
- Coordinated Universal Time;
- Gregorian Calendar.

1.3.3 Abbreviations defined in ISO 1000

This part of ISO 10303 makes use of the following abbreviation defined in ISO 1000:

- SI International System of Units.

Section 2 : Generic product description resources

2.1 Introduction

This section specifies the ISO 10303 integrated resources used for the high-level description of a product. The following schemas are included:

- **application_context_schema**;
- **product_definition_schema**;
- **product_property_definition_schema**;
- **product_property_representation_schema**.

The **application_context_schema** supports the description of the applicable usage of product data.

The **product_definition_schema** supports the description of the identification of products, the categorization of products, and the relationships among the definitions of products.

The **product_property_definition_schema** supports the description of the characteristics of a product.

The **product_property_representation_schema** describes the structure for representing the shape characteristics of a product.

NOTE 1 – This part is not limited to the domain of a specific application context. The reader should not be misled by the deliberately limited scope of the examples.

NOTE 2 – The way in which the support resources are associated with the generic product description resources is described in annex E.

2.2 application_context_schema

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

EXPRESS specification:

```
*)
SCHEMA application_context_schema;

REFERENCE FROM support_resource_schema
    (label,
     text);
```

```
REFERENCE FROM date_time_schema
   (year_number);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema	clause 4.13 of this part of ISO 10303
date_time_schema	clause 4.9 of this part of ISO 10303

2 – See annex F, figure F.1, for a graphical presentation of this schema.

3 – This schema is one of the generic product description resource schemas. The relationships between the generic product description resource schemas are given in annex D.

2.2.1 Introduction

This clause defines requirements for the **application_context_schema**. This schema defines a mechanism for application protocols to define a frame of reference or context that applies to particular sets of product data. An overall application context has several context elements. Each context element may be referenced by a different aspect of product data.

A means is provided for users of product data to identify the application interpreted model in terms of which that product data is defined.

2.2.2 Fundamental concepts and assumptions

The meaningful exchange of product data requires the identification of the application context in which that product data is defined. One element of an application context is the application interpreted model which is used. Each application interpreted model belongs to one, and only one, application protocol. Since all application interpreted models in ISO 10303 have unique names, identification of an application interpreted model schema name also uniquely identifies an application protocol. Applications which use product data require information that identifies the application interpreted model that defines the content and logical structure of the product data.

2.2.3 application_context_schema entity definitions

2.2.3.1 application_context

An **application_context** is a context in which product data is defined. An **application_context** represents various types of information which relate to product data and may affect the meaning and usage of that data.

EXPRESS specification:

```
*)
ENTITY application_context;
```

```
  application          : text;
INVERSE
  context_elements : SET [1:?] OF  application_context_element
                                FOR frame_of_reference;
END_ENTITY;
(*
```

Attribute definitions:

application: a description of the usage of the product data.

context_elements: there shall be at least one **application_context_element** associated with the **application_context**.

2.2.3.2 application_protocol_definition

An **application_protocol_definition** is a definition of an application protocol.

EXPRESS specification:

```
*)
ENTITY application_protocol_definition;
  status                                    : label;
  application_interpreted_model_schema_name : label;
  application_protocol_year                 : year_number;
  application                               : application_context;
END_ENTITY;
(*
```

Attribute definitions:

status: a classification of an application protocol with regard to its acceptance by some approving body.

NOTES

1 – Legal values for this attribute are specified in application interpreted models which use this entity.

2 – The fundamental assumption (see clause 2.2.2) that an application interpreted model schema name uniquely identifies an application protocol is only valid for ISO 10303 application protocols.

application_interpreted_model_schema_name: the EXPRESS schema name of the the application interpreted model.

application_protocol_year: the year when the application protocol attained the status given by the **status** attribute.

application: the application context of the application protocol.

2.2.3.3 application_context_element

An **application_context_element** is an aspect of the application context in which product data is defined. **Application_context_elements** may represent several different aspects of an application context. An application protocol may specify the context of data using the subtypes of the **application_context_element** entity:

- **product_context**;
- **product_definition_context**;
- **product_concept_context**;
- **library_context**.

NOTE – In using this entity, it is the responsibility of an application protocol to properly constrain the values of these attributes based on their applicable usage.

EXAMPLE 1 – A **product_definition_context** could have "design" as its **product_life_cycle_stage** and "detailed design" as its **application_context_element.name**.

EXPRESS specification:

```
*)
ENTITY application_context_element
  SUPERTYPE OF (ONEOF (product_context,
                       product_definition_context,
                       product_concept_context,
                       library_context));
  name                 : label;
  frame_of_reference   : application_context;
END_ENTITY;
(*
```

Attribute definitions:

name: an identification of the detailed context in which product data exists.

EXAMPLE 2 – "Detailed design" and "preliminary design" are examples of **application_context_element.names**.

frame_of_reference: the overall context of which this **application_context_element** is a part.

2.2.3.4 product_context

A **product_context** is that aspect of an **application_context** which defines a context for a **product**.

NOTE 1 – The **product** entity is defined in clause 2.3.4.1 in this part of ISO 10303.

A **product_context** represents information about the engineering or manufacturing perspective for product data that may affect its meaning and usage.

EXPRESS specification:

```
*)
ENTITY product_context
  SUBTYPE OF (application_context_element);
  discipline_type : label;
END_ENTITY;
(*
```

Attribute definitions:

discipline_type: an identification of the engineering or manufacturing category to which the **product** belongs.

EXAMPLE 3 – "Electrical", "mechanical", and "architectural" are examples of **discipline_types**.

NOTE 2 – Legal values for this attribute are specified in application interpreted models which use this entity.

2.2.3.5 product_definition_context

A **product_definition_context** is that aspect of an **application_context** which defines a context for a **product_definition**.

NOTE 1 – The **product_definition** entity is defined in clause 2.3.4.8 of this part of ISO 10303.

This context represents information about the stage in the product life cycle to which the product data belongs. Such information may affect the meaning and usage of the product data.

EXPRESS specification:

```
*)
ENTITY product_definition_context
  SUBTYPE OF (application_context_element);
  life_cycle_stage : label;
END_ENTITY;
(*
```

Attribute definitions:

life_cycle_stage: an identification of the general stage in the product life cycle to which the product data belongs.

NOTE 2 – Legal values for this attribute are specified in application interpreted models which use this entity.

EXAMPLE 4 – Terms such as "as designed" and "as specified" are examples of **life_cycle_stages**.

2.2.3.6 product_concept_context

A **product_concept_context** is that aspect of an **application_context** which defines a context for a **product_concept**.

NOTE – The **product_concept** entity is defined in ISO 10303-44.

A **product_concept_context** represents information that relates to the characterization of potential purchasers of a product. Such information may affect the meaning and usage of the product data.

EXPRESS specification:

```
*)
ENTITY product_concept_context
  SUBTYPE OF (application_context_element);
  market_segment_type : label;
END_ENTITY;
(*
```

market_segment_type: an identification of the category that characterizes potential purchasers of a product.

EXAMPLE 5 – "Luxury automobiles", "laptop personal computers", and "budget personal stereos" are all examples of **market_segment_types**.

2.2.3.7 library_context

A **library_context** is that aspect of an **application_context** which defines a context for elements of a library. A **library_context** represents information that may affect the meaning and usage of the product data in the library.

EXPRESS specification:

```
*)
ENTITY library_context
  SUBTYPE OF (application_context_element);
  library_reference : label;
END_ENTITY;
(*
```

Attribute definitions:

library_reference: an identification of the library which provides a context for the elements of the library.

EXAMPLE 6 – "Smith's Pens (UK) Standard Pen Tops" is an example of a **library_reference**.

EXPRESS specification:

```
*)
END_SCHEMA; -- application_context_schema
(*
```

2.3 product_definition_schema

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

EXPRESS specification:

```
*)
SCHEMA product_definition_schema;

REFERENCE FROM application_context_schema
    (product_context,
     product_definition_context);

REFERENCE FROM document_schema
    (document);

REFERENCE FROM effectivity_schema
    (effectivity);

REFERENCE FROM support_resource_schema
    (bag_to_set,
     identifier,
     label,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

application_context_schema	clause 2.2 of this part of ISO 10303
document_schema	clause 4.2 of this part of ISO 10303
effectivity_schema	clause 4.11 of this part of ISO 10303
support_resource_schema	clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.2, for a graphical presentation of this schema.

3 – This schema is one of the generic product description resource schemas. The relationships between the generic product description resource schemas are given in annex D.

2.3.1 Introduction

This clause defines requirements for the **product_definition_schema.** This schema provides for the generic aspects of product definition.

EXAMPLE 7 – The identification of products and definitions of products, the grouping of products according to classification schemes and the definition of various kinds of relationship between products are all examples of generic aspects of product definition.

The generic aspects of product definition are realized by means of a set of resource constructs that may be used to specify generic product definition facts.

EXAMPLE 8 – The fact that a product definition is related to other product definitions is an example of generic product definition fact. It is not specific to a particular application context.

Product definition facts are independent of properties. The ways in which these facts may be combined are prescribed by the relationships that are defined in this schema. Each generic fact can be interpreted in any application context; the ways in which generic facts are to be interpreted are stipulated in application protocols.

NOTE – An example of the way in which this schema may be used is given in annex E.

2.3.2 Fundamental concepts and assumptions

A single product may have multiple groups of definitions associated with it; each group is valid in a given application context.

EXAMPLE 9 – An application context may be the manufacture of ball-point pens. A particular ball-point pen has multiple versions. Each version is described through a group of product definitions. An older version has a solid cap whereas a more recent version has a hole in the cap to prevent people from choking if they swallow it. The characteristics of the two versions would be different from each other because the later version would have a hole in the cap whereas the earlier version would not. This schema could be used to define the ball-point pen with a solid cap as one product and the one with a hole in the cap as another. On the other hand it could also be used to define both pens as two versions of a single product. The approach taken would depend upon the requirements of the application context.

The definition of a product includes the properties that are required to characterize it.

EXAMPLE 10 – An integrated circuit product will have a functional definition represented by a circuit schematic diagram and a physical definition represented by a circuit layout diagram.

Definitions of products may be related to each other in various ways. It is necessary to be able to identify the relationships between products and to be able to characterize those relationships.

EXAMPLE 11 – The ball-point pen cap would be related in an assembly relationship to the ball-point pen itself and one of the caps could be substituted for the other in some circumstances.

2.3.3 product_definition_schema type definition: source

The **source** type represents whether a **product_definition_formation** is to be manufactured within an organization or bought-in. Possible values are constrained to represent these two possibilities, or the fact that this information is not known.

EXPRESS specification:

```
*)
TYPE source = ENUMERATION OF
  (made,
   bought,
   not_known);
END_TYPE;
(*
```

2.3.4 product_definition_schema entity definitions

2.3.4.1 product

A **product** is the identification and description, in an application context, of a physically realizable object that is produced by a process.

NOTES

1 – A physically realizable object may, but need not, be physically realized.

2 – The term "product" is defined in ISO 10303-1.

EXAMPLE 12 – Production, construction, manufacture, and fabrication are all examples of processes.

EXAMPLE 13 – The ball-point pen, its cap, and the assembly of the cap and the ball-point pen are all objects that may be produced by a manufacturing process.

EXAMPLE 14 – The end-product that an organization produces and intends to sell, the raw material from which such products are manufactured, and the tooling and other facilities that are used to transform the raw material into the end-product may all be examples of physically realizable objects.

EXPRESS specification:

```
*)
ENTITY product;
  id                 : identifier;
  name               : label;
  description        : text;
  frame_of_reference : SET [1:?] OF product_context;
UNIQUE
  UR1: id;
END_ENTITY;
(*
```

Attribute definitions:

id: the identification of the **product**.

EXAMPLE 15 – Part numbers, stock item numbers and serial numbers are examples of **product** identifiers.

name: the word or group of words by which the **product** is referred to.

EXAMPLE 16 – "Ball-point pen", "cap", and "nib" are examples of **product names**.

description: text that relates the nature of the **product**.

frame_of_reference: the contexts within which the **product** was defined.

Formal propositions:

UR1: every **product**'s identification shall be unique.

2.3.4.2 product_category

A **product_category** identifies a category to which **products** and other **product_categorys** may belong.

EXAMPLE 17 – In an application protocol whose context includes manufactured parts, "mechanical part", "electrical part", "structural part", "piping part", "water pipe", and "hot water pipe" are all examples of **product_categorys**.

EXPRESS specification:

```
*)
ENTITY product_category;
  name        : label;
  description : OPTIONAL text;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **product_category** is referred to.

description: text that relates the nature of the **product_category**.

Informal propositions:

IP1: when a **product_category** participates in a **product_category_relationship**, the value of the **name** attribute shall be unique within its parent **product_categorys**.

2.3.4.3 product_related_product_category

A **product_related_product_category** is a **product_category** that contains **products**.

EXPRESS specification:

```
*)
ENTITY product_related_product_category
  SUBTYPE OF (product_category);
  products : SET [1:?] OF product;
END_ENTITY;
(*
```

Attribute definitions:

products: the **products** that belong to the **product_related_product_category**.

2.3.4.4 product_category_relationship

A **product_category_relationship** is an association between two categories. A **product_category** may be identified as a sub-category in its relationship with other **product_categorys**.

EXAMPLE 18 – The **product_category** called "piping part" could be used to relate two other **product_categorys**, "cold water pipe" and "hot water pipe", to each other.

Networks of **product_category**s may be defined using this entity.

EXAMPLE 19 – A **product_category** called "piping part" may be the parent category of sub-categories called "cold water pipe" and "hot water pipe". The **product_category** called "hot water pipe" may be a sub-category of another **product_category** called "boiler output pipe". This is an example of a **product_category** network because the **product_category** called "hot water pipe" has two parents.

If a **product** is within a **product_category**, it is also within all of the parent **product_category**s of that **product_category**.

NOTE – This entity, in conjunction with the **product_category** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY product_category_relationship;
  name          : label;
  description   : text;
  category      : product_category;
  sub_category  : product_category;
WHERE
  WR1: acyclic_product_category_relationship (SELF, [SELF.sub_category]);
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **product_category_relationship** is referred to.

description: text that relates the nature of the **product_category_relationship**.

category: the parent of the **sub_category**.

EXAMPLE 20 – In the previous example, "piping part" would be the **category** for "cold water pipe" and "hot water pipe".

sub_category: a child of the **category**.

EXAMPLE 21 – In the previous example "cold water pipe" would be the **sub_category** for "piping part" in one case and "hot water pipe" would be the **sub_category** in the other.

Within a single category and sub-category relationship hierarchy, a given **product** shall only be referenced once.

Formal propositions:

WR1: a network of **product_category**s shall not be cyclic.

Informal propositions:

IP1: a given **product** may only be referenced once within a given **product_category_relationship** structure.

2.3.4.5 product_definition_formation

A **product_definition_formation** is an identified group of **product_definitions** for a **product**. Each of the **product_definitions** in the group has a distinct **frame_of_reference** (and therefore **life_cycle_stage**) or includes some variation in its properties.

NOTE – The purpose of the **product_definition_formation** is defined in application interpreted models that use this entity.

EXAMPLE 22 – An application interpreted model could use this entity to support the identification of different versions of a single **product**. Each version would be described by a unique group of **product_definitions** and each group, identified by a **product_definition_formation**, would be associated with the same **product**.

EXPRESS specification:

```
*)
ENTITY product_definition_formation;
  id          : identifier;
  description : text;
  of_product  : product;
UNIQUE
  UR1: id, of_product;
END_ENTITY;
(*
```

Attribute definitions:

id: the unique identification of the **product_definition_formation** in the context of the **product** to which it relates.

EXAMPLE 23 – Part version number is an example of a **product_definition_formation** identifier.

description: text that relates the nature of the **product_definition_formation**.

NOTE 1 – The descriptions of different formations of a single **product** could identify differences in the purpose and function of each formation.

of_product: the **product** of which the **product_definition_formation** is a version.

NOTE 2 – A **product** is associated with one or more **product_definition_formations** through the implicit inverse of this relationship.

Formal propositions:

UR1: the **id** of each **product_definition_formation** that is related to a single **product** (through their **of_product** attributes) shall be unique within the collection of **product_definition_formations** which are related to that **product**.

2.3.4.6 product_definition_formation_relationship

A **product_definition_formation_relationship** is an association between two **product_definition_formations**. An association may exist between **product_definition_formations**

that relate to different **product**s or between different formations of the same **product**. The meaning of the relationship for a particular context is defined in specializations of this entity.

NOTES

1 – Relationships captured using this entity may be parent child relationships. Specializations of this entity state this fact if it is true for the particular specialization.

2 – This entity, in conjunction with the **product_definition_formation** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY product_definition_formation_relationship;
  id                                    : identifier;
  name                                  : label;
  description                           : text;
  relating_product_definition_formation : product_definition_formation;
  related_product_definition_formation  : product_definition_formation;
END_ENTITY;
(*
```

Attribute definitions:

id: the identification of the **product_definition_formation_relationship**.

name: the word or group of words by which the **product_definition_formation_relationship** is referred to.

description: text that relates the nature of the **product_definition_formation_relationship**.

relating_product_definition_formation: one of the **product_definition_formation**s which is a part of the relationship.

NOTE 3 – The role of this attribute is defined in the application protocol or the ISO 10303 integrated resource that uses or specializes this entity.

related_product_definition_formation: the other **product_definition_formation** which is a part of the relationship. If one element of the relationship is dependent upon the other then this attribute shall be the dependent one.

NOTE 4 – The role of this attribute is defined in the application protocol or the ISO 10303 integrated resource that uses or specializes this entity.

2.3.4.7 product_definition_formation_with_specified_source

A **product_definition_formation_with_specified_source** is a **product_definition_formation** with a specified source.

EXPRESS specification:

```
*)
ENTITY product_definition_formation_with_specified_source
```

```
  SUBTYPE OF (product_definition_formation);
  make_or_buy : source;
END_ENTITY;
(*
```

Attribute definitions:

make_or_buy: the **source** of the **product_definition_formation**.

2.3.4.8 product_definition

A **product_definition** is the identification of a characterization of a product in a particular application context.

NOTE – A **product_definition** is characterized by properties which refer to it.

EXAMPLE 24 – A **product**'s physical design may be one **product_definition** while the functional design of the same **product** may be a different **product_definition**. Both **product_definitions** would be related to the same **product_definition_formation** but could be used in different application contexts.

EXPRESS specification:

```
*)
ENTITY product_definition;
  id                 : identifier;
  description        : text;
  formation          : product_definition_formation;
  frame_of_reference : product_definition_context;
END_ENTITY;
(*
```

Attribute definitions:

id: the identification of the **product_definition**.

description: text that relates the nature of the **product_definition**.

formation: the **product_definition_formation** to which the **product_definition** relates.

frame_of_reference: the **product_definition_context** in which the **product_definition** or **product_definition** data is used.

2.3.4.9 product_definition_with_associated_documents

A **product_definition_with_associated_documents** is a **product_definition** which is related to one or more **documents**.

EXPRESS specification:

```
*)
ENTITY product_definition_with_associated_documents
  SUBTYPE OF(product_definition);
```

```
  documentation_ids : SET[1:?] OF document;
END_ENTITY;
(*
```

Attribute definitions:

documentation_ids: the set of **documents** that are associated with the **product_definition.**

2.3.4.10 product_definition_relationship

A **product_definition_relationship** is an association between two **product_definitions**. An association may exist between **product_definitions** that relate to different **products** or between different definitions of the same **product.**

EXAMPLE 25 – The relationships within a bill-of-materials structure are examples of **product_definition_relationships** that associate different **products**. The relationship between a sketch and a detailed design is an example of a **product_definition_relationship** that associates different definitions of a single **product**.

A single **product_definition** may be used more than once within the description of a **product.**

NOTE 1 – The same component could be used more than once in the same assembly. Each usage of the component would be specified as an instance of the **product_definition_relationship** entity.

The meaning of the relationship for a particular context is defined in specializations of this entity.

NOTES

2 – Relationships captured using this entity may be parent-child relationships. Specializations of this entity state this fact if it is true for the particular specialization.

3 – This entity, in conjunction with the **product_definition** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY product_definition_relationship;
  id                          : identifier;
  name                        : label;
  description                 : text;
  relating_product_definition : product_definition;
  related_product_definition  : product_definition;
END_ENTITY;
(*
```

Attribute definitions:

id: the identification of the **product_definition_relationship.**

name: the word or group of words by which the **product_definition_relationship** is referred to.

description: text that relates the nature of the **product_definition_relationship.**

relating_product_definition: one of the **product_definitions** which is a part of the relationship.

NOTE 4 – The role of this attribute is defined in the application protocol or the ISO 10303 integrated resource that uses or specializes this entity.

EXAMPLE 26 – If the **product_definition_relationship** is an assembly component relationship, the **relating_product_definition** may be the assembly.

related_product_definition: the other **product_definition** which is a part of the relationship. If one element of the relationship is dependent upon the other, this attribute shall be the dependent one.

NOTE 5 – The role of this attribute is defined in the application protocol or the ISO 10303 integrated resource that uses or specializes this entity.

EXAMPLE 27 – In an assembly, the **related_product_definition** may be the **product_definition** that is an element of the assembly.

2.3.4.11 product_definition_substitute

A **product_definition_substitute** is an association between three **product_definitions** where one **product_definition** acts as a substitute for another **product_definition** in the context of a third **product_definition**.

NOTE – If a **product_definition_relationship** exists between a component part and its assembly, a **product_definition_substitute** could be used to capture an allowable substitute, that is, a different component part that may be used as a replacement in the same assembly.

EXAMPLE 28 – Two kinds of ball-point pen may be manufactured: a standard model and a deluxe model. Each model of the pen would be specified as a separate **product_definition** related to a particular kind of nib: a standard nib and a deluxe nib, respectively. The fact that the deluxe nib and the standard nib are interchangeable only in the context of the standard model of ball-point pen can be established with this construct.

EXPRESS specification:

```
*)
ENTITY product_definition_substitute;
  description          : text;
  context_relationship  : product_definition_relationship;
  substitute_definition : product_definition;
WHERE
  WR1: context_relationship.related_product_definition :<>: substitute_definition;
END_ENTITY;
(*
```

Attribute definitions:

description: text that relates the nature of the **product_definition_substitute**.

context_relationship: the relationship whose **related_definition** may be replaced by the **substitute_definition**. The **relating_product_definition** attribute of this attribute specifies the context within which the replacement is made.

EXAMPLE 29 – The **relating_product_definition** of a **product_definition_relationship** identified as a **context_relationship** would be the standard model of the ball-point pen and the **related_product_definition** would be the standard nib.

substitute_definition: the **product_definition** which may act as a replacement for the **related_product_definition** of the **context_relationship**.

EXAMPLE 30 – This attribute would be the deluxe nib in the previous examples.

Formal propositions:

WR1: a **product_definition** shall not be defined as a substitute for itself.

2.3.4.12 product_definition_effectivity

A **product_definition_effectivity** is an **effectivity** that may be applied for a particular **product_definition** participating in a given **product_definition_relationship**.

EXAMPLE 31 – The fact that a particular **product_definition**'s shape was effective for a certain period of time within the context of an assembly could be defined using this entity. The **product_definition_effectivity** and **dated_effectivity** entities would be associated with the **product_definition_shape** entity during interpretation. The usage attribute of the **product_definition_effectivity** entity would represent the assembly.

EXPRESS specification:

```
*)
ENTITY product_definition_effectivity
  SUBTYPE OF (effectivity);
  usage : product_definition_relationship;
UNIQUE
  UR1: usage, SELF\effectivity.id;
END_ENTITY;
(*
```

Attribute definitions:

usage: the **product_definition_relationship** that initiates the **effectivity**.

Formal propositions:

UR1: the identification of the **effectivity** shall be unique.

2.3.5 product_definition_schema function definitions

2.3.5.1 acyclic_product_definition_formation_relationship

The **acyclic_product_definition_formation_relationship** function determines whether or not the given **product_definition_formations** have been self-defined by the associations made

in the specified **product_definition_formation_relationship**. This function may be used to evaluate either a **product_definition_formation_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **product_definition_formation_relationship** entity is either a **product_definition_formation_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **product_definition_formation_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_product_definition_formation_relationship
  (relation          : product_definition_formation_relationship;
   relatives         : SET OF product_definition_formation;
   specific_relation : STRING) : LOGICAL;

  LOCAL
    x               : SET OF product_definition_formation_relationship;
    i               : INTEGER;
    local_relatives : SET OF product_definition_formation;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_product_definition_formation :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_product_definition_formation, specific_relation));
       local_relatives := relatives + relation.relating_product_definition_formation;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_product_definition_formation_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **product_definition_formation_relationship** to be checked.

relatives: (input) the set of **product_definition_formations** which the function is searching for in the **relating_product_definition_formation** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **product_definition_formation_relationship** entity.

2.3.5.2 acyclic_product_definition_relationship

The **acyclic_product_definition_relationship** function determines whether or not the given **product_definitions** have been self-defined by the associations made in the specified **product_definition_relationship**. This function may be used to evaluate either a **product_definition_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **product_definition_relationship** entity is either a **product_definition_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **product_definition_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_product_definition_relationship
  (relation          : product_definition_relationship;
   relatives         : SET OF product_definition;
   specific_relation : STRING) : LOGICAL;

  LOCAL
    x               : SET OF product_definition_relationship;
    i               : INTEGER;
    local_relatives : SET OF product_definition;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_product_definition :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_product_definition, specific_relation));
       local_relatives := relatives + relation.relating_product_definition;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_product_definition_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
```

```
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **product_definition_relationship** to be checked.

relatives: (input) the set of **product_definition**s which the function is searching for in the **relating_product_definition** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **product_definition_relationship** entity.

2.3.5.3 acyclic_product_category_relationship

The **acyclic_product_category_relationship** function determines whether or not the given **product_category**s have been self-defined by the associations made in the specified **product_category_relationship**.

The function returns a value of TRUE if none of the elements of the **children** argument occur in the **relation** argument. Otherwise it returns a value of FALSE.

NOTE – This function is based on the cyclicity avoidance template that is described in annex D. It performs its task by ensuring that the **children** argument is not the **category** field of either the **relation** argument or, recursively, any other **product_category_relationship** entities in which the **relation** plays the role of **category**.

EXPRESS specification:

```
*)
FUNCTION acyclic_product_category_relationship
  (relation : product_category_relationship;
   children : SET of product_category): LOGICAL;

  LOCAL
    x              : SET OF product_category_relationship;
    i              : INTEGER;
    local_children : SET OF product_category;
  END_LOCAL;

  REPEAT i := 1 to HIINDEX(children);
    If relation.category :=: children [i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.category,
```

```
                          'PRODUCT_DEFINITION_SCHEMA.' +
                          'PRODUCT_CATEGORY_RELATIONSHIP.SUB_CATEGORY'));
  local_children := children + relation.category;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 to HIINDEX(x);
      IF NOT acyclic_product_category_relationship(x[i], local_children) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **product_category_relationship** to be checked.

children: (input) the set of **product_category** which the function is searching for in the **category** field of the **relation** argument.

EXPRESS specification:

```
*)
END_SCHEMA; -- product_definition_schema
(*
```

2.4 product_property_definition_schema

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

EXPRESS specification:

```
*)
SCHEMA product_property_definition_schema;

REFERENCE FROM product_definition_schema
    (product_definition,
     product_definition_relationship);

REFERENCE FROM support_resource_schema
    (bag_to_set,
     label,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

product_definition_schema	clause 2.3 of this part of ISO 10303
support_resource_schema	clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.3, for a graphical presentation of this schema.

3 – This schema is one of the generic product description resource schemas. The relationships between the generic product description resource schemas are given in annex D.

2.4.1 Introduction

This clause defines requirements for the **product_property_definition_schema**. This schema brings together groups of characteristics. These characteristics may be associated with a single **product_definition** or with one **product_definition** in the context of another. Each characteristic is independent of the number or types of representations of that characteristic. These characteristics may also be associated with the shape of a product, an element of the shape of a product, or sometimes the relationship between elements of the shape of a product.

2.4.2 Fundamental concepts and assumptions

Products are characterized by their properties. This schema allow these properties to be captured and to be associated to the identification of a **product_definition**, **product_definition_relationship** or a **shape_definition**. The properties are identifiable concepts which are independent of how, or even if, they are represented.

EXAMPLE 32 – The definition of the surface finish of a shape is independent of the way in which the shape or the surface finish are represented.

NOTE – In the previous example, both the defined surface finish and the shape could have multiple representations.

It is appropriate to specify entities which carry the idea of properties as concepts that are independent of their representation.

2.4.3 product_property_definition_schema type definitions

2.4.3.1 characterized_definition

The **characterized_definition** type is the means by which the various properties that pertain to a **characterized_object**, **product_definition** or **shape_definition** are aggregated. This type is used to allow properties to be associated with such entities.

EXPRESS specification:

```
*)
TYPE characterized_definition = SELECT
  (characterized_object,
```

```
   characterized_product_definition,
   shape_definition);
END_TYPE;
(*
```

2.4.3.2 characterized_product_definition

The **characterized_product_definition** type is the means by which the various properties that pertain to a specific **product_definition** are aggregated. This type is used to allow properties to be associated with such entities.

There is a special meaning in the selection of a **product_definition_relationship**; a property which is related to a **product_definition_relationship** is applied to the **related_product_definition** attribute in the context of its **relating_product_definition** attribute.

NOTE 1 – This allows product properties which change when the product is a part of the relationship to be represented.

EXAMPLE 33 – The shape of a gasket depends upon whether or not it is a part of an assembly and, if it is an element of an assembly, it depends upon the assembly that it participates in.

NOTE 2 – A **characterized_product_definition** refers to a **product_definition** either directly or in the context of another **product_definition** through a **product_definition_relationship**.

EXPRESS specification:

```
*)
TYPE characterized_product_definition = SELECT
  (product_definition,
   product_definition_relationship);
END_TYPE;
(*
```

2.4.3.3 shape_definition

The **shape_definition** type provides a mechanism by which either the entire shape of a **product**, a limited portion of the shape of a **product**, or an association between portions of the shapes of **products** can be referenced.

NOTE – Unlike the **characterized_product_definition** type, references to a **shape_aspect_relationship** are references to the association itself and not one of the associated **shape_aspects**.

EXPRESS specification:

```
*)
TYPE shape_definition = SELECT
  (product_definition_shape,
   shape_aspect,
   shape_aspect_relationship);
END_TYPE;
(*
```

2.4.4 product_property_definition_schema entity definitions

2.4.4.1 characterized_object

A **characterized_object** is the identification of an item that has associated property information.

NOTES

1 – A **characterized_object** is characterized by properties which refer to it.

2 – The properties of a **characterized_object** may be used as an environmental condition under which the properties of a product are measured.

EXAMPLE 34 – If a product has a set of properties that are measured within a room or an atmosphere, the properties of the room or atmosphere, such as temperature, are specified with the attribute that references this entity. The name and description of the room or the atmosphere are captured through the **characterized_object** entity.

EXPRESS specification:

```
*)
ENTITY characterized_object;
  name        : label;
  description : text;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **characterized_object** is referred to.

description: text that relates the nature of the **characterized_object**.

2.4.4.2 property_definition

A **property_definition** is a property that characterizes a product. It applies to either a product, a product in the context of another product, the shape of a product, an element of the shape of a product, or an element of the shape in the context of another element of the shape of a product.

EXPRESS specification:

```
*)
ENTITY property_definition;
  name        : label;
  description : text;
  definition  : characterized_definition;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **property_definition** is referred to.

description: text that relates the nature of the **property_definition**.

definition: the **product_definition**, **shape_aspect** or **shape_aspect_relationship** whose property is identified.

NOTE – A **property_definition** cannot exist unless it is related to either a **product_definition** or a **shape_definition**. This attribute establishes this relationship.

2.4.4.3 product_definition_shape

A **product_definition_shape** is a type of **property_definition**. It is the shape of a product. This shape may be a conceptual shape for which a specific geometric representation is not required.

NOTE – Early in the design of a **product** there may not be a specific idea about the form of the **product** but there may be certain characteristics of the form that are to be captured. Those **product** shape characteristics can be attached to the **product** shape using this entity.

EXAMPLE 35 – A geometric representation of shape is not needed to assert facts such as, "a shape must fit within a 5 centimetre cube".

EXPRESS specification:

```
*)
ENTITY product_definition_shape
  SUBTYPE OF (property_definition);
UNIQUE
  UR1: SELF\property_definition.definition;
WHERE
  WR1: 'PRODUCT_PROPERTY_DEFINITION_SCHEMA.CHARACTERIZED_PRODUCT_DEFINITION'
       IN TYPEOF (SELF\property_definition.definition);
END_ENTITY;
(*
```

Formal propositions:

UR1: each **characterized_product_definition** shall be related to zero or one **product_definition_shapes**.

WR1: the **definition** attribute shall be a **characterized_product_definition.**

2.4.4.4 shape_aspect

A **shape_aspect** is an identifiable element of the shape of a **product.**

EXAMPLE 36 – Consider the **product_definition_shape** of a bolt. One might distinguish, as an element of this shape, the concept of the threaded portion of its shank. This portion of the shape could be specified using a **shape_aspect** entity so that other properties, such as surface finish, may be associated with it.

EXPRESS specification:

```
*)
ENTITY shape_aspect;
  name                 : label;
  description          : text;
  of_shape             : product_definition_shape;
  product_definitional : LOGICAL;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **shape_aspect** is referred to.

description: text that relates the nature of the **shape_aspect.**

of_shape: the **product_definition_shape** of which this entity is an aspect.

EXAMPLE 37 – If the identified aspect were the threaded portion of a bolt's shank, this attribute would be the the **product_definition_shape** of the bolt.

product_definitional: an indication that the **shape_aspect** is on the physical boundary of the **product_definition_shape.** If the value of this attribute is TRUE, it is asserted that the **shape_aspect** being identified is on such a boundary. If the value is FALSE, it shall be asserted that the **shape_aspect** being identified is not on such a boundary. If the value is UNKNOWN, it shall be asserted that it is not known whether or not the **shape_aspect** being identified is on such a boundary.

EXAMPLE 38 – If the identified **shape_aspect** were the threaded portion of a bolt's shank, the value of this attribute would be TRUE. However, if it were the centre-line, the value would be FALSE.

2.4.4.5 shape_aspect_relationship

A **shape_aspect_relationship** is an association between two **shape_aspects.**

EXAMPLE 39 – If one **shape_aspect** is part of another, this entity could be used to associate the two **shape_aspects.**

NOTES

1 – The intention is to capture distinct properties of shapes as they exist in their own right and as they participate in the relationship. The **shape_aspect_relationship** entity is not a representation

definition entity. If a **shape_aspect_relationship** is to have an explicit representation then that is a separate concept which is described using resource constructs that are outside this schema.

2 – Each **shape_aspect** entity could have different properties.

3 – No geometric relationship is established between related **shape_aspect** entities.

EXAMPLE 40 – A **shape_aspect_relationship** might relate two **shape_aspects** whose representations are the equivalent surfaces of a mould and a moulded product. The shape of the mould is not spatially related to the moulded product.

NOTES

4 – Relationships captured using this entity may be parent-child relationships. Specializations of this entity state this fact if it is true for the particular specialization.

5 – This entity, in conjunction with the **shape_aspect** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY shape_aspect_relationship;
  name                 : label;
  description          : text;
  relating_shape_aspect : shape_aspect;
  related_shape_aspect  : shape_aspect;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **shape_aspect_relationship** is referred to.

description: text that relates the nature of the **shape_aspect_relationship**.

relating_shape_aspect: one of the **shape_aspect** entities which is a part of the relationship.

EXAMPLE 41 – A **shape_aspect** that is a pocket with five faces would play the role of **relating_shape_aspect** in five **shape_aspect_relationship** entities: one per face.

related_shape_aspect: the other **shape_aspect** entity which is a part of the relationship. If one element of the relationship is dependent upon the other, this attribute shall be the dependent one.

EXAMPLE 42 – In the previous example each of the five **shape_aspect_relationship** entities would have a different **shape_aspect** entity in the **related_shape_aspect** field. There would be one for each side and one for the bottom of the pocket.

2.4.5 product_property_definition_schema function definition: acyclic_shape_aspect_relationship

The **acyclic_shape_aspect_relationship** function determines whether or not the given **shape_aspects** have been self-defined by the associations made in the specified **shape_aspect_relationship**. This function may be used to evaluate either a **shape_aspect_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **shape_aspect_relationship** entity is either a **shape_aspect_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **shape_aspect_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_shape_aspect_relationship
  (relation          : shape_aspect_relationship;
   relatives         : SET OF shape_aspect;
   specific_relation : STRING) : LOGICAL;

  LOCAL
    x               : SET OF shape_aspect_relationship;
    i               : INTEGER;
    local_relatives : SET OF shape_definition;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_shape_aspect :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_shape_aspect, specific_relation));
       local_relatives := relatives + relation.relating_shape_aspect;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_shape_aspect_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **shape_aspect_relationship** to be checked.

relatives: (input) the set of **shape_aspect**s which the function is searching for in the **relating_shape_aspect** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **shape_aspect_relationship** entity.

EXPRESS specification:

```
*)
END_SCHEMA; -- product_property_definition_schema
(*
```

2.5 product_property_representation_schema

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

EXPRESS specification:

```
*)
SCHEMA product_property_representation_schema;

REFERENCE FROM product_property_definition_schema
    (shape_definition,
     property_definition,
     product_definition_shape);

REFERENCE FROM representation_schema
    (representation,
     representation_relationship);

REFERENCE FROM product_definition_schema
    (product_definition,
     product_definition_relationship);

REFERENCE FROM support_resource_schema
    (bag_to_set);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

product_property_representation_schema	clause 2.4 of this part of ISO 10303
representation_schema	ISO 10303-43
product_definition_schema	clause 2.3 of this part of ISO 10303
support_resource_schema	clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.4, for a graphical presentation of this schema.

3 – This schema is one of the generic product description resource schemas. The relationships between the generic product description resource schemas are given in annex D.

2.5.1 Introduction

This clause defines requirements for the **product_property_representation_schema**. This schema allows for the representation of a product's properties.

NOTE 1 – This schema is only used to represent a product's shape property.

The requirements addressed by this schema are:

– the ability to identify multiple representations, founding various types of **geometric_representation_items**, as representing a **shape_definition**;

NOTE 2 – The verb "to found" and the **geometric_representation_item** entity are defined in ISO 10303-43.

– the ability to represent the shape of zero, one, or more **product_definition_shapes**;

– the ability to identify various representations as representing the shape of zero, one, or more **shape_aspects**;

– the ability to identify relationships between the representations of shapes and to state that the identified relationships are the representations of zero, one or more **product_definition** entities.

2.5.2 Fundamental concepts and assumptions

The following assumptions were made in preparing this schema:

a) ISO 10303 will include various types of representations ranging from simple collections of geometry to more elaborate collections of representations of different kinds of properties;

b) it is possible to have more than one representation of a single property;

c) a single representation may be used to represent a property that relates to zero, one, or many **product_definitions**;

d) any property of a **product_definition** is an identifiable concept independent of how, or even if, it is represented.

NOTE – The different representation schemes are defined in other parts of ISO 10303. This schema provides a structure within which these different resources can be brought together to represent the properties of a product.

2.5.3 product_property_representation_schema entity definitions

2.5.3.1 shape_representation

A **shape_representation** is a specific kind of **representation** that represents a shape.

EXPRESS specification:

```
*)
ENTITY shape_representation
  SUBTYPE OF (representation);
END_ENTITY;
(*
```

2.5.3.2 property_definition_representation

A **property_definition_representation** is an association between a property of a product and its representation.

EXPRESS specification:

```
*)
ENTITY property_definition_representation;
  definition          : property_definition;
  used_representation : representation;
END_ENTITY;
(*
```

Attribute definitions:

definition: the identification of the property that is being represented.

used_representation: the representation of the property which is defined by the **definition** attribute.

2.5.3.3 shape_representation_relationship

A **shape_representation_relationship** is an association between two **representations** where at least one of the **representations** is a **shape_representation**.

EXAMPLE 43 – The **representation** of the shape of a bolt may be related to the **representation** of a position when the bolt is a part of an assembly.

EXPRESS specification:

```
*)
ENTITY shape_representation_relationship
  SUBTYPE OF (representation_relationship);
WHERE
  WR1: 'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.SHAPE_REPRESENTATION' IN
       (TYPEOF(SELF\representation_relationship.rep_1) +
        TYPEOF(SELF\representation_relationship.rep_2));
END_ENTITY;
(*
```

Formal propositions:

WR1: one of the two representations in the **shape_representation_relationship** shall be a **shape_representation**.

2.5.3.4 context_dependent_shape_representation

A **context_dependent_shape_representation** is the **representation** of the shape of a **product_definition** in the context of a role that it plays in a **product_definition_relationship**.

EXAMPLE 44 – A gasket's shape depends upon whether the gasket is participating in an assembly relationship and, if it is a part of an assembly, the product or products that it is assembled with.

EXAMPLE 45 – The position of a bolt's shape depends upon the way it is positioned in a given assembly.

In such cases, the representation of the product's shape must be specified in the context of the representation which influences it. This entity establishes a relationship between a **shape_representation** and a **product_definition** as it plays the role of **related_product_definition** in a **product_definition_relationship**.

EXPRESS specification:

```
*)
ENTITY context_dependent_shape_representation;
  representation_relation      : shape_representation_relationship;
  represented_product_relation : product_definition_shape;
WHERE
  WR1: 'PRODUCT_DEFINITION_SCHEMA.PRODUCT_DEFINITION_RELATIONSHIP'
       IN TYPEOF (SELF.represented_product_relation.definition);
END_ENTITY;
(*
```

Attribute definitions:

representation_relation: the relationship between the representations of the shapes of the the **related_product_definition** and the **relating_product_definition** of the **represented_product_relation**.

represented_product_relation: the **product_definition_shape** that is the definition of the shape of the **product_definition** that plays the role of **related_product_definition** in a

product_definition_relationship.

Formal propositions:

WR1: the **context_dependent_shape_representation** shall represent the **related_product_definition** in the **product_definition_relationship** entity.

2.5.3.5 shape_definition_representation

A **shape_definition_representation** is the representation of a **shape_definition.**

EXPRESS specification:

```
*)
ENTITY shape_definition_representation
  SUBTYPE OF (property_definition_representation);
WHERE
  WR1: ('PRODUCT_PROPERTY_DEFINITION_SCHEMA.SHAPE_DEFINITION' IN
       TYPEOF (SELF.definition.definition))
       OR
       ('PRODUCT_PROPERTY_DEFINITION_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN
       TYPEOF (SELF.definition));
  WR2: 'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.SHAPE_REPRESENTATION' IN
       TYPEOF(SELF.used_representation);
END_ENTITY;
(*
```

Formal propositions:

WR1: the inherited **representation_of** attribute shall be a **shape_definition.**

WR2: the inherited **representation_model** attribute shall be a **shape_representation.**

2.5.4 product_property_representation_schema function definitions

2.5.4.1 relatives_of_product_definitions

The **relatives_of_product_definitions** function finds all of the **product_definitions** which are related to one or more elements of the **definition_set** argument. Only those relationships which are established by the subtype of the **product_definition_relationship** entity which is given in the **relation_subtype** argument are considered by this function.

EXPRESS specification:

```
*)
FUNCTION relatives_of_product_definitions
```

```
   (definition_set   : SET OF product_definition;
    relation_subtype : STRING) : SET OF product_definition;

  FUNCTION local_relatives_of_product_definitions
     (definition_set    : SET OF product_definition;
      total_definitions : SET OF product_definition;
      relation_subtype  : STRING) : SET OF product_definition;
    LOCAL
      i           : INTEGER;
      local_def   : SET OF product_definition := [];
      local_pdr   : SET OF product_definition_relationship := [];
      local_total : SET OF product_definition := [];
   END_LOCAL;

    REPEAT i := 1 TO HIINDEX(definition_set);
      local_pdr := local_pdr +
                   bag_to_set(USEDIN
                              (definition_set[i],
                               relation_subtype + '.RELATING_PRODUCT_DEFINITION'));
    END_REPEAT;

    REPEAT i := 1 TO HIINDEX(local_pdr);
      local_def := local_def + local_pdr[i].related_product_definition;
    END_REPEAT;

    IF (SIZEOF(local_def) - SIZEOF(total_definitions)) = 0 THEN
      RETURN (local_def);
    ELSE
      local_total := total_definitions + local_def;
      RETURN(local_def +
             (local_relatives_of_product_definitions
              (local_def - total_definitions, local_total, relation_subtype)));
    END_IF;
  END_FUNCTION;

  RETURN (local_relatives_of_product_definitions
          (definition_set, definition_set, relation_subtype));

END_FUNCTION;
(*
```

Argument definitions:

definition_set: (input) the set of **product_definitions** to be operated on.

relation_subtype: the fully qualified name of an entity which is a subtype of the **product_definition_relationship** entity.

2.5.4.2 relatives_of_shape_representations

The **relatives_of_shape_representation** function finds all of the **shape_representations** which are related to one or more elements of the **shape_rep_set** argument.

EXPRESS specification:

```
*)
FUNCTION relatives_of_shape_representations
   (shape_representation_set : SET OF shape_representation) :
    SET OF shape_representation;

  FUNCTION local_relatives_of_shape_representations
     (shape_representation_set : SET OF shape_representation;
      total_reps    : SET OF shape_representation) : SET OF shape_representation;

    LOCAL
      i               : INTEGER;
      local_shape_rep : SET OF shape_representation := [];
      local_srr       : SET OF shape_representation_relationship := [];
      local_total     : SET OF shape_representation := [];
    END_LOCAL;

    REPEAT i := 1 TO HIINDEX(shape_representation_set);
      local_srr := local_srr + bag_to_set(USEDIN(shape_representation_set[i],
                   'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.'+
                   'SHAPE_REPRESENTATION_RELATIONSHIP.REP_1'));
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(local_srr);
      local_shape_rep := local_shape_rep + local_srr[i].rep_2;
    END_REPEAT;
    IF SIZEOF (local_shape_rep - total_reps) = 0 THEN
      RETURN (shape_representation_set);
    ELSE
      local_total := total_reps + local_shape_rep;
      RETURN(local_shape_rep + (local_relatives_of_shape_representations
                               (local_shape_rep - total_reps, local_total)));
    END_IF;
  END_FUNCTION;

  RETURN (local_relatives_of_shape_representations
        (shape_representation_set, shape_representation_set));

  END_FUNCTION;
(*
```

Argument definitions:

shape_representation_set: (input) the set of **shape_representations** to be operated on.

EXPRESS specification:

```
*)
END_SCHEMA; -- product_property_representation_schema
(*
```

Section 3 : Management resources

3.1 Introduction

This section specifies structures which allow administrative data to be associated with product data in specific application contexts. It includes the following schema:

– **management_resources_schema**.

The **management_resources_schema** contains the structures which allow administrative data to be associated with product data in specific application contexts.

3.2 management_resources_schema

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

EXPRESS specification:

```
*)
SCHEMA management_resources_schema;

REFERENCE FROM support_resource_schema
    (label);

REFERENCE FROM application_context_schema
    (library_context);

REFERENCE FROM document_schema;

REFERENCE FROM action_schema;

REFERENCE FROM certification_schema;

REFERENCE FROM approval_schema;

REFERENCE FROM contract_schema;

REFERENCE FROM security_classification_schema;

REFERENCE FROM person_organization_schema;

REFERENCE FROM date_time_schema;

REFERENCE FROM group_schema;
```

```
REFERENCE FROM effectivity_schema;
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema	clause 4.13 of this part of ISO 10303
application_context_schema	clause 2.2 of this part of ISO 10303
document_schema	clause 4.2 of this part of ISO 10303
action_schema	clause 4.3 of this part of ISO 10303
certification_schema	clause 4.4 of this part of ISO 10303
approval_schema	clause 4.5 of this part of ISO 10303
contract_schema	clause 4.6 of this part of ISO 10303
security_classification_schema	clause 4.7 of this part of ISO 10303
person_organization_schema	clause 4.8 of this part of ISO 10303
date_time_schema	clause 4.9 of this part of ISO 10303
group_schema	clause 4.10 of this part of ISO 10303
effectivity_schema	clause 4.11 of this part of ISO 10303

2 – See annex F, figure F.5, for a graphical presentation of this schema.

3 – This schema contains generic management resources which conform to the template in annex E. The way in which generic management resources are used is also described in annex E.

3.2.1 Introduction

This clause defines requirements for the **management_resources_schema**. The EXPRESS constructs in this schema allow management type data to be associated with other aspects of product data in specific application contexts.

3.2.2 Fundamental concepts and assumptions

The relationship between management type data and other aspects of product data is application-specific. For this reason it must be possible to associate management type data with other aspects of product data in specific application contexts.

3.2.3 management_resources_schema entity definitions

3.2.3.1 name_assignment

A **name_assignment** is an association of a **name** with product data.

EXPRESS specification:

```
*)
ENTITY name_assignment
  ABSTRACT SUPERTYPE;
  assigned_name : label;
END_ENTITY;
```

```
(*
```

Attribute definitions:

assigned_name: the word, or group of words, which is to be associated with the product data.

3.2.3.2 external_referent_assignment

An **external_referent_assignment** is an identification of an entity in an application context that may be referenced from outside sources.

A unique name is assigned to the entity to facilitate referencing. An ISO 10303 integrated resource may specify generic characteristics of the name and an application protocol may specify application-specific characteristics or legal values of the name.

EXPRESS specification:

```
*)
ENTITY external_referent_assignment
  ABSTRACT SUPERTYPE;
  assigned_name : label;
UNIQUE
  UR1 : assigned_name;
END_ENTITY;
(*
```

Attribute definitions:

assigned_name: a unique **label** that the entity may be referenced by.

Formal propositions:

UR1: the **assigned_name** shall be unique.

3.2.3.3 library_assignment

A **library_assignment** is an **external_referent_assignment** which is part of a **library_context**.

EXPRESS specification:

```
*)
ENTITY library_assignment
  ABSTRACT SUPERTYPE
  SUBTYPE OF (external_referent_assignment);
  frame_of_reference : library_context;
UNIQUE
  UR1: frame_of_reference;
END_ENTITY;
(*
```

Attribute definitions:

frame_of_reference: the context in which the **library_assignment** is defined.

Formal propositions:

UR1: the **frame_of_reference** shall be unique.

3.2.3.4 document_reference

A **document_reference** is an association of a **document** with product data.

NOTE – The concept of **document** is described in clause 4.2.

EXPRESS specification:

```
*)
ENTITY document_reference
  ABSTRACT SUPERTYPE;
  assigned_document : document;
  source            : label;
END_ENTITY;
(*
```

Attribute definitions:

assigned_document: the **document** which is to be associated with the product data.

source: the identification of where the **assigned_document** is from.

EXAMPLE 46 – "Engineering" or "library" are examples of sources.

3.2.3.5 action_request_assignment

An **action_request_assignment** is an association of a **versioned_action_request** with product data.

NOTE – The concept of **versioned_action_request** is described in clause 4.3.

EXPRESS specification:

```
*)
ENTITY action_request_assignment
  ABSTRACT SUPERTYPE;
  assigned_action_request : versioned_action_request;
END_ENTITY;
(*
```

Attribute definitions:

assigned_action_request: the **versioned_action_request** which is to be associated with the product data.

3.2.3.6 action_assignment

An **action_assignment** is an association of an **action** with product data.

NOTE – The concept of **action** is described in clause 4.3.

EXPRESS specification:

```
*)
ENTITY action_assignment
  ABSTRACT SUPERTYPE;
  assigned_action : action;
END_ENTITY;
(*
```

Attribute definitions:

assigned_action: the **action** which is to be associated with the product data.

3.2.3.7 certification_assignment

A **certification_assignment** is an association of a **certification** with product data.

NOTE – The concept of **certification** is described in clause 4.4.

EXPRESS specification:

```
*)
ENTITY certification_assignment
  ABSTRACT SUPERTYPE;
  assigned_certification : certification;
END_ENTITY;
(*
```

Attribute definitions:

assigned_certification: the **certification** which is to be associated with the product data.

3.2.3.8 approval_assignment

A **approval_assignment** is an association of a **approval** with product data.

NOTE – The concept of **approval** is described in clause 4.5.

EXPRESS specification:

```
*)
ENTITY approval_assignment
  ABSTRACT SUPERTYPE;
  assigned_approval : approval;
END_ENTITY;
(*
```

Attribute definitions:

assigned_approval: the **approval** which is to be associated with the product data.

3.2.3.9 contract_assignment

A **contract_assignment** is an association of a **contract** with product data.

NOTE – The concept of **contract** is described in clause 4.6.

EXPRESS specification:

```
*)
ENTITY contract_assignment
  ABSTRACT SUPERTYPE;
  assigned_contract : contract;
END_ENTITY;
(*
```

Attribute definitions:

assigned_contract: the **contract** which is to be associated with the product data.

3.2.3.10 security_classification_assignment

A **security_classification_assignment** is an association of a **security_classification** with product data.

NOTE – The concept of **security_classification** is described in clause 4.7.

EXPRESS specification:

```
*)
ENTITY security_classification_assignment
  ABSTRACT SUPERTYPE;
  assigned_security_classification : security_classification;
END_ENTITY;
(*
```

Attribute definitions:

assigned_security_classification: the **security_classification** which is to be associated with the product data.

3.2.3.11 person_assignment

A **person_assignment** is an association of a **person** with product data.

NOTE – The concept of **person** is described in clause 4.8.

EXPRESS specification:

```
*)
ENTITY person_assignment
  ABSTRACT SUPERTYPE;
  assigned_person : person;
  role            : person_role;
END_ENTITY;
(*
```

Attribute definitions:

assigned_person: the **person** which is to be associated with the product data.

role: the function performed by the **assigned_person**.

3.2.3.12 organization_assignment

A **organization_assignment** is an association of a **organization** with product data.

NOTE – The concept of **organization** is described in clause 4.8.

EXPRESS specification:

```
*)
ENTITY organization_assignment
  ABSTRACT SUPERTYPE;
  assigned_organization : organization;
  role                  : organization_role;
END_ENTITY;
(*
```

Attribute definitions:

assigned_organization: the **organization** which is to be associated with the product data.

role: the function performed by the **assigned_organization**.

3.2.3.13 person_and_organization_assignment

A **person_and_organization_assignment** is an association of a **person_and_organization** with product data.

NOTE – The concept of **person_and_organization** is described in clause 4.8.

EXPRESS specification:

```
*)
ENTITY person_and_organization_assignment
  ABSTRACT SUPERTYPE;
  assigned_person_and_organization : person_and_organization;
```

```
  role                                : person_and_organization_role;
END_ENTITY;
(*
```

Attribute definitions:

assigned_person_and_organization: the **person_and_organization** which is to be associated with the product data.

role: the function performed by the **assigned_person_and_organization**.

3.2.3.14 date_assignment

A **date_assignment** is an association of a **date** with product data.

NOTE – The concept of **date** is described in clause 4.9.

EXPRESS specification:

```
*)
ENTITY date_assignment
  ABSTRACT SUPERTYPE;
  assigned_date : date;
  role          : date_role;
END_ENTITY;
(*
```

Attribute definitions:

assigned_date: the **date** which is to be associated with the product data.

role: the function performed by the **assigned_date**.

3.2.3.15 time_assignment

A **time_assignment** is an association of a **time** with product data.

NOTE – The concept of **time** is described in clause 4.9.

EXPRESS specification:

```
*)
ENTITY time_assignment
  ABSTRACT SUPERTYPE;
  assigned_time : local_time;
  role          : time_role;
END_ENTITY;
(*
```

Attribute definitions:

assigned_time: the **time** which is to be associated with the product data.

role: the function performed by the **assigned_time**.

3.2.3.16 date_and_time_assignment

A **date_and_time_assignment** is an association of a **date_and_time** with product data.

NOTE – The concept of **date_and_time** is described in clause 4.9.

EXPRESS specification:

```
*)
ENTITY date_and_time_assignment
  ABSTRACT SUPERTYPE;
  assigned_date_and_time : date_and_time;
  role                   : date_time_role;
END_ENTITY;
(*
```

Attribute definitions:

assigned_date_and_time: the **date_and_time** which is to be associated with the product data.

role: the function performed by the **assigned_date_and_time**.

3.2.3.17 group_assignment

A **group_assignment** is an association of a **group** with product data.

NOTE – The concept of **group** is described in clause 4.10.

EXPRESS specification:

```
*)
ENTITY group_assignment
  ABSTRACT SUPERTYPE;
  assigned_group : group;
END_ENTITY;
(*
```

Attribute definitions:

assigned_group: the **group** which is to be associated with the product data.

3.2.3.18 effectivity_assignment

A **effectivity_assignment** is an association of a **effectivity** with product data.

NOTE – The concept of **effectivity** is described in clause 4.11.

EXPRESS specification:

```
*)
ENTITY effectivity_assignment
  ABSTRACT SUPERTYPE;
  assigned_effectivity : effectivity;
END_ENTITY;
(*
```

Attribute definitions:

assigned_effectivity: the **effectivity** which is to be associated with the product data.

EXPRESS specification:

```
*)
END_SCHEMA; -- management_resources_schema
(*
```

Section 4 : Support resources

4.1 Introduction

This section specifies the ISO 10303 integrated resources that are used by several other ISO 10303 integrated resource schemas. The following schemas are included:

- **document_schema**;
- **action_schema**;
- **certification_schema**;
- **approval_schema**;
- **contract_schema**;
- **security_classification_schema**;
- **person_organization_schema**;
- **date_time_schema**;
- **group_schema**;
- **effectivity_schema**;
- **external_reference_schema**;
- **support_resource_schema**;
- **measure_schema**.

The **document_schema** allows references to documents outside ISO 10303 to be defined.

The **action_schema** allows action, requests for action, and the status of actions to be defined.

The **certification_schema** allows documents attesting to facts to be referred to.

The **approval_schema** allows authorization data to be described.

The **contract_schema** allows agreements to be referred to.

The **security_classification_schema** allows degrees of secrecy to be specified.

The **person_organization_schema** allows information that identifies people and organizations to be defined.

The **date_time_schema** allows dates and times to be specified.

The **group_schema** allows groups of things to be defined.

The **effectivity_schema** allows the effectivity of usages of product data to be defined.

The **external_reference_schema** allows references to information outside a given application protocol's domain.

The **support_resource_schema** provides basic data types.

The **measure_schema** allows physical quantities to be defined.

NOTE – The way in which the support resources are associated with the generic product description resources is described in annex E.

4.2 document_schema

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

EXPRESS specification:

```
*)
SCHEMA document_schema;

REFERENCE FROM support_resource_schema
    (bag_to_set,
     identifier,
     label,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.6, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.2.1 Introduction

This clause defines requirements for the **document_schema**. The resource constructs in this schema enable the description of citations of formal standards or documents that are outside the domain of ISO 10303. These resource constructs shall be used to reference additional information that is relevant to the description of the product but not as the product data itself.

EXAMPLE 47 – International, national, and organizational standards, catalogues, and tables of engineering data are examples of formal standards or documents.

NOTE – The use of the resource constructs in this schema should be unambiguous and should contain enough information to allow users (people or software) to access the relevant information.

4.2.2 Fundamental concepts and assumptions

Product data can include citations of other information sources.

EXAMPLE 48 – Heat treatment processes cannot be defined using the integrated product description resource but they can be specified using the resource constructs that are defined in this schema.

4.2.3 document_schema entity definitions

4.2.3.1 document_type

A **document_type** is the sort of data that the formal standards or documents are being used to describe in a particular application context.

EXAMPLE 49 – "Material", "surface finish", and "heat treatment process" are all pieces of data that can be described implicitly, by reference to other documents (such as DIN documents), rather than explicitly every time they are used.

EXPRESS specification:

```
*)
ENTITY document_type;
  product_data_type : label;
END_ENTITY;
(*
```

Attribute definitions:

product_data_type: the name of the sort of data that the document is being used to describe.

EXAMPLE 50 – AISI 303 is a material description document.

4.2.3.2 document

A **document** is an unambiguous reference to a formal standard or document that is defined outside ISO 10303.

EXPRESS specification:

```
*)
ENTITY document;
  id          : identifier;
  name        : label;
  description : text;
  kind        : document_type;
UNIQUE
  UR1: id;
END_ENTITY;
(*
```

Attribute definitions:

id: an identification of the **document**.

name: the word or group of words by which the **document** is referred to.

NOTE – The **name** may include the source of the **document**, such as, ISO or DIN.

description: text that relates the nature of the **document**.

kind: the sort of data that the **document** describes.

Formal propositions:

UR1: the **id** shall be unique.

4.2.3.3 document_with_class

A **document_with_class** is a **document** which identifies different classifications for the data which it describes.

EXAMPLE 51 – A surface finish document may be identified by various classes, for example, class A, class B, and class C. Each class of surface finish specifies various allowances for imperfections in the surface finish. For example, class A may require no visible imperfections, class B may require no more than two imperfections that are more than 0.06 inch diameter within any given square inch of surface area, and class C may require no imperfections under 10 times magnification.

EXPRESS specification:

```
*)
ENTITY document_with_class
  SUBTYPE OF (document);
  class : identifier;
END_ENTITY;
(*
```

Attribute definitions:

class: the identification of the data classification that is being referenced.

NOTE – The value of this attribute would be 'A', 'B' or 'C' in the previous example.

4.2.3.4 document_usage_constraint

A **document_usage_constraint** identifies a specific subject area or aspect from within a **document** and gives the relevant information or text which applies. The semantics of the reference can be found in the **document** itself.

EXAMPLE 52 – This entity could be used to define the clauses of a document that are relevant.

EXPRESS specification:

```
*)
ENTITY document_usage_constraint;
  source                : document;
  subject_element       : label;
  subject_element_value : text;
END_ENTITY;
(*
```

Attribute definitions:

source: the identification of where the **document** originates.

NOTE – The inverse of this relationship is used to define multiple elements of a **document**.

subject_element: the name of one element of the **source**.

subject_element_value: a specific value from the **source**.

EXAMPLE 53 – For references to a "surface finish" document the **subject_element** may be a "surface_imperfection" and its associated **subject_element_value** may be "no visible imperfections", "no more than two imperfections that are more than 0.06 inch diameter within any given square inch of surface area", or "no imperfections under 10 times magnification".

4.2.3.5 document_relationship

A **document_relationship** is an association between two **documents**.

NOTE 1 – Relationships captured using this entity may be parent child relationships. Specializations of this entity state this fact if it is true for the particular specialization.

NOTE 2 – This entity, in conjunction with the **document** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY document_relationship;
  name                : label;
  description         : text;
  relating_document   : document;
  related_document    : document;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **document_relationship** is referred to.

description: text that relates the nature of the **document_relationship**.

relating_document: one of the **documents** which is a part of the relationship.

related_document: the other **document** entity which is a part of the relationship. If one element of the relationship is dependent upon the other then this attribute shall be the dependent one.

4.2.4 document_schema function definition: acyclic_document_relationship

The **acyclic_document_relationship** function determines whether or not the given **documents** have been self-defined by the associations made in the specified **document_relationship**. This function may be used to evaluate either a **document_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **document_relationship** entity is either a **document_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **document_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_document_relationship
   (relation          : document_relationship;
    relatives         : SET OF document;
    specific_relation : STRING) : LOGICAL;

  LOCAL
    x               : SET OF document_relationship;
    i               : INTEGER;
    local_relatives : SET OF document;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_document :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_document, specific_relation));
  local_relatives := relatives + relation.relating_document;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_document_relationship
          (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **document_relationship** to be checked.

relatives: (input) the set of **document**s which the function is searching for in the **relating_document** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **document_relationship** entity.

EXPRESS specification:

```
*)
END_SCHEMA; -- document_schema
(*
```

4.3 action_schema

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

EXPRESS specification:

```
*)
SCHEMA action_schema;

REFERENCE FROM support_resource_schema
    (bag_to_set,
     identifier,
     label,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.7, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.3.1 Introduction

This clause defines requirements for the **action_schema**. A particular set of product data is often the result of changes to previous versions of that data. Changes can occur for many reasons.

EXAMPLE 54 – Reasons for action include evolving user requirements, manufacturing problems and difficulties when a product is in use.

This schema allows for actions which have caused change, and the reasons for those actions, to be described.

4.3.2 Fundamental concepts and assumptions

Action information can be attached to any aspect of a product data.

NOTE – Histories of changes are not supported in this edition of this International Standard.

4.3.3 action_schema type definition: supported_item

The **supported_item** type represents the use to which an **action_resource** may be put.

EXPRESS specification:

```
*)
TYPE supported_item = SELECT
  (action_directive,
   action,
   action_method);
END_TYPE;
(*
```

4.3.4 action_schema entity definitions

4.3.4.1 action

An **action** is the effort made to realize a specific result.

EXPRESS specification:

```
*)
ENTITY action;
  name          : label;
  description   : text;
  chosen_method : action_method;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **action** is referred to.

description: text that relates the nature of the **action**.

chosen_method: the procedure used to carry out the **action**.

4.3.4.2 executed_action

An **action_execution** is an **action** which has been carried out.

EXPRESS specification:

```
*)
ENTITY executed_action
  SUBTYPE OF (action);
END_ENTITY;
(*
```

4.3.4.3 directed_action

A **directed_action** is an **executed_action** which has been carried out as a result of a directive.

EXPRESS specification:

```
*)
ENTITY directed_action
  SUBTYPE OF (executed_action);
directive : action_directive;
END_ENTITY;
(*
```

Attribute definitions:

directive: the **action_directive** against which the **directed_action** was made.

4.3.4.4 action_status

An **action_status** is a ranking which indicates the state of an **executed_action**.

EXPRESS specification:

```
*)
ENTITY action_status;
  status          : label;
  assigned_action : executed_action;
END_ENTITY;
(*
```

Attribute definitions:

status: a description of the ranking.

assigned_action: the **action** to which the **status** applies.

4.3.4.5 action_request_status

An **action_request_status** is the ranking which gives an indication of the state of a request for action.

EXPRESS specification:

```
*)
ENTITY action_request_status;
  status           : label;
  assigned_request : versioned_action_request;
END_ENTITY;
(*
```

Attribute definitions:

status: a description of the ranking.

assigned_request: the **versioned_action_request** to which the **status** applies.

4.3.4.6 action_relationship

An **action_relationship** is an association between two **actions**.

NOTE – This entity, in conjunction with the **action** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY action_relationship;
  name            : label;
  description     : text;
  relating_action : action;
  related_action  : action;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **action_relationship** is referred to.

description: text that relates the nature of the **action_relationship**.

relating_action: one of the related **actions**.

related_action: the other related **action**. If one element of the relationship is dependent upon the other then this attribute shall be the dependent one.

4.3.4.7 action_method

An **action_method** is a potential means of satisfying the requirements that are highlighted in a **versioned_action_request**.

EXPRESS specification:

```
*)
ENTITY action_method;
  name        : label;
```

```
  description : text;
  consequence : text;
  purpose     : text;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **action_method** is referred to.

description: text that relates the nature of the **action_method**.

consequence: an informal description of the effects of the **action_method**.

purpose: an informal description of the rationale behind the **action_method**.

4.3.4.8 action_request_solution

An **action_request_solution** identifies the way in which the requirements that are highlighted in a **versioned_action_request** were satisfied.

EXPRESS specification:

```
*)
ENTITY action_request_solution;
  method  : action_method;
  request : versioned_action_request;
END_ENTITY;
(*
```

method: the way in which the requirements that are highlighted in the **request** were satisfied.

request: the requirements that are satisfied by the **method**.

4.3.4.9 action_method_relationship

An **action_method_relationship** is an association between two **action_methods**.

NOTE – This entity, in conjunction with the **action_method** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY action_method_relationship;
  name            : label;
  description     : text;
  relating_method : action_method;
  related_method  : action_method;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **action_method_relationship** is referred to.

description: text that relates the nature of the **action_method_relationship**.

relating_method: one of the related **action_methods**.

related_method: the other related **action_method**. If one element of the relationship is dependent upon the other then this attribute shall be the dependent one.

4.3.4.10 versioned_action_request

A **versioned_action_request** is a formal notification of a desire for action to be taken.

EXPRESS specification:

```
*)
ENTITY versioned_action_request;
  id          : identifier;
  version     : label;
  purpose     : text;
  description : text;
END_ENTITY;
(*
```

Attribute definitions:

id: the means of identification of the **versioned_action_request**.

version: the identification of the version of the **versioned_action_request**.

purpose: an informal description of the reason for the **versioned_action_request**.

description: an informal definition of the **versioned_action_request**.

4.3.4.11 action_directive

An **action_directive** is the formal notification that authority has been given to perform an **action**.

NOTE – The distinction between a **versioned_action_request** and an **action_directive** is the level of authority that is associated with it. Anyone can submit a **versioned_action_request** whereas only authorized people or organizations can submit **action_directives** that are to be acted upon. A request asks for action whereas a directive demands action.

EXPRESS specification:

```
*)
ENTITY action_directive;
  name        : label;
  description : text;
  analysis    : text;
  comment     : text;
  requests    : SET[1:?] OF versioned_action_request;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **action_directive** is referred to.

description: text that relates the nature of the **action_directive**.

analysis: an informal description of the results of the analysis that was carried out on the elements of the **requests** set.

EXAMPLE 55 – The fact that two different requests are asking for the same effect could be recorded in this attribute.

comment: an informal description of any other pertinent information.

requests: the **versioned_action_requests** to which the **action_directive** relates.

4.3.4.12 action_resource

An **action_resource** is the means which are used to carry out an **action_directive**.

EXPRESS specification:

```
*)
ENTITY action_resource;
  name         : label;
  description  : text;
  usage        : SET [1:?] OF supported_item;
  kind         : action_resource_type;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **action_resource** is referred to.

description: text that relates the nature of the **action_resource**.

usage: the way in which the **action_resource** is used.

kind: the sort of **action_resource** that is being used.

4.3.4.13 action_resource_relationship

An **action_resource_relationship** is an association between two **action_resources**.

NOTE – This entity, in conjunction with the **action_resource** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY action_resource_relationship;
  name              : label;
  description       : text;
  relating_resource : action_resource;
```

```
  related_resource  : action_resource;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **action_resource_relationship** is referred to.

description: text that relates the nature of the **action_resource_relationship**.

relating_resource: one of the related **action_resources**.

related_resource: the other related **action_resource**. If one element of the relationship is dependent upon the other then this attribute shall be the dependent one.

4.3.4.14 action_resource_type

An **action_resource_type** is the type of means which are used to carry out an **action_directive**.

EXPRESS specification:

```
*)
ENTITY action_resource_type;
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **action_resource_type** is referred to.

4.3.5 action_schema function definitions

4.3.5.1 acyclic_action_relationship

The **acyclic_action_relationship** function determines whether or not the given **actions** have been self-defined by the associations made in the specified **action_relationship**. This function may be used to evaluate either a **action_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **action_relationship** entity is either an **action_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **action_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_action_relationship
         (relation          : action_relationship;
          relatives         : SET [1:?] OF action;
          specific_relation : STRING) : LOGICAL;
  LOCAL
    x               : SET [1:?] OF action_relationship;
    i               : INTEGER;
    local_relatives : SET [1:?] OF action;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_action :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_action, specific_relation));
  local_relatives := relatives + relation.relating_action;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_action_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **action_relationship** to be checked.

relatives: (input) the set of **actions** which the function is searching for in the **relating_action** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **action_relationship** entity.

4.3.5.2 acyclic_action_resource_relationship

The **acyclic_action_resource_relationship** function determines whether or not the given **action_resources** have been self-defined by the associations made in the specified **action_resource_relationship**. This function may be used to evaluate either a **action_resource_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **action_resource_relationship** entity is either an **action_resource_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **action_resource_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_action_resource_relationship
         (relation          : action_resource_relationship;
          relatives         : SET [1:?] OF action_resource;
          specific_relation : STRING) : LOGICAL;
  LOCAL
    x               : SET [1:?] OF action_resource_relationship;
    i               : INTEGER;
    local_relatives : SET [1:?] OF action_resource;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_resource :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_resource, specific_relation));
  local_relatives := relatives + relation.relating_resource;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_action_resource_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **action_resource_relationship** to be checked.

relatives: (input) the set of **action_resources** which the function is searching for in the **relating_action_resource** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **action_resource_relationship** entity.

4.3.5.3 acyclic_action_method_relationship

The **acyclic_action_method_relationship** function determines whether or not the given **action_methods** have been self-defined by the associations made in the specified **action_method_relationship**. This function may be used to evaluate either a **action_method_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **action_method_relationship** entity is either an **action_method_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **action_method_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_action_method_relationship
         (relation          : action_method_relationship;
          relatives         : SET [1:?] OF action_method;
          specific_relation : STRING) : LOGICAL;
  LOCAL
    x              : SET [1:?] OF action_method_relationship;
    i              : INTEGER;
    local_relatives : SET [1:?] OF action_method;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_method :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_method, specific_relation));
  local_relatives := relatives + relation.relating_method;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_action_method_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
```

```
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **action_method_relationship** to be checked.

relatives: (input) the set of **action_method**s which the function is searching for in the **relating_action_method** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **action_method_relationship** entity.

EXPRESS specification:

```
*)
END_SCHEMA; -- action_schema
(*
```

4.4 certification_schema

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

EXPRESS specification:

```
*)
SCHEMA certification_schema;

REFERENCE FROM support_resource_schema
    (label,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.8, for a graphical presentation of this schema.

3 This schema contains support resources.

4.4.1 Introduction

This clause defines requirements for the **certification_schema**. Certification assures and validates product data.

EXAMPLE 56 – A material certificate states the chemical composition of one or more physical pieces of material, for example, purchased raw material. The presence of the material certificate removes the need to test the composition of the material; it allows the specified material composition to be accepted as fact without further investigation.

EXAMPLE 57 – A certified supplier can supply goods that do not require checking.

4.4.2 Fundamental concepts and assumptions

Certification information can be attached to any aspect of product data.

4.4.3 certification_schema entity definitions

4.4.3.1 certification_type

A **certification_type** is the kind of thing that is being certified.

EXAMPLE 58 – Suppliers and manufacturers can be certified.

EXPRESS specification:

```
*)
ENTITY certification_type;
  description : label;
END_ENTITY;
(*
```

Attribute definitions:

description: a description of the thing that is being certified.

EXAMPLE 59 – The value of this attribute could be "supplier" or "manufacturer".

4.4.3.2 certification

Certification is documentation which asserts facts.

Application protocols that use this resource construct shall specify the extent of the certification.

EXPRESS specification:

```
*)
ENTITY certification;
  name    : label;
  purpose : text;
  kind    : certification_type;
```

```
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **certification** is referred to.

purpose: an informal description of the reason why the **certification** has been applied.

EXAMPLE 60 – A purpose for material certification could be the fact that certain customers demand material certification for the products that are supplied to them.

kind: the sort of **certification** that is being applied.

EXPRESS specification:

```
*)
END_SCHEMA; -- certification_schema
(*
```

4.5 approval_schema

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

EXPRESS specification:

```
*)
SCHEMA approval_schema;

REFERENCE FROM support_resource_schema
    (bag_to_set,
     label,
     text);

REFERENCE FROM date_time_schema
    (date_time_select);

REFERENCE FROM person_organization_schema
    (person_organization_select);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema	clause 4.13 of this part of ISO 10303
date_time_schema	clause 4.9 of this part of ISO 10303
person_organization_schema	clause 4.8 of this part of ISO 10303

2 – See annex F, figure F.9, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.5.1 Introduction

This clause defines requirements for the **approval_schema**. Approval information concerns the acceptance of product data.

EXAMPLE 61 – One version of a product may be approved for manufacture whilst another may still be undergoing design.

4.5.2 Fundamental concepts and assumptions

Approval information can be attached to any aspect of product data.

NOTE – Histories of approvals are not supported in this International Standard.

4.5.3 approval_schema entity definitions

4.5.3.1 approval_status

An **approval_status** is the ranking that gives an indication of the state of an **approval**.

EXAMPLE 62 – "Approved" and "disapproved" are examples of **approval_statuses**.

EXPRESS specification:

```
*)
ENTITY approval_status;
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the name of the ranking of an **approval**.

4.5.3.2 approval

An **approval** is a confirmation of the quality of the product data that it is related to.

Application protocols that use this resource construct shall specify the extent of the approval.

EXPRESS specification:

```
*)
ENTITY approval;
  status : approval_status;
  level  : label;
END_ENTITY;
(*
```

Attribute definitions:

status: the status of the **approval** in terms of whether or not the **approval** has been granted.

level: the type or level of **approval** in terms of its usage. This usage may be implied rather than explicit.

EXAMPLE 63 – One possible level of **approval** is "released for production"; this explicitly identifies the approved usage. Another possible level is "preliminary design completed"; this only implies the approved usage which will depend upon company-specific procedures.

4.5.3.3 approval_date_time

An **approval_date_time** is an association between an **approval** and a date and/or time.

EXPRESS specification:

```
*)
ENTITY approval_date_time;
  date_time      : date_time_select;
  dated_approval : approval;
END_ENTITY;
(*
```

Attribute definitions:

date_time: the moment when the **dated_approval** is given.

dated_approval: the **approval** with which the date and/or time is associated.

4.5.3.4 approval_person_organization

An **approval_person_organization** is an association between an **approval** and a given person and/or organization in a given role.

EXPRESS specification:

```
*)
ENTITY approval_person_organization;
  person_organization : person_organization_select;
  authorized_approval : approval;
  role                : approval_role;
END_ENTITY;
(*
```

Attribute definitions:

person_organization: the person and/or organization playing the given role.

authorized_approval: the **approval** that is effected by the **person_organization**.

role: the function performed by the **person_organization**.

4.5.3.5 approval_role

An **approval_role** is a function performed with respect to an **approval**.

EXPRESS specification:

```
*)
ENTITY approval_role;
  role : label;
END_ENTITY;
(*
```

Attribute definitions:

role: the name of the performed function.

4.5.3.6 approval_relationship

An **approval_relationship** is an association between two **approvals**.

NOTE – This entity, in conjunction with the **approval** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY approval_relationship;
  name              : label;
  description       : text;
  relating_approval : approval;
  related_approval  : approval;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **approval_relationship** is referred to.

description: text that relates the nature of the **approval_relationship**.

relating_approval: one of the **approvals** that is a part of the association.

related_approval: the other **approval** that is a part of the association. If one element of the relationship is dependent upon the other, this attribute shall be the dependent one.

4.5.4 approval_schema function definition: acyclic_approval_relationship

The **acyclic_approval_relationship** function determines whether or not the given **approvals** have been self-defined by the associations made in the specified **approval_relationship**. This function may be used to evaluate either a **approval_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **approval_relationship** entity is either an **approval_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **approval_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_approval_relationship
   (relation          : approval_relationship;
    relatives         : SET OF approval;
    specific_relation : STRING) : LOGICAL;

  LOCAL
    x              : SET OF approval_relationship;
    i              : INTEGER;
    local_relatives : SET OF approval;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_approval :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_approval, specific_relation));
  local_relatives := relatives + relation.relating_approval;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_approval_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```


Argument definitions:

relation: (input) the candidate **approval_relationship** to be checked.

relatives: (input) the set of **approvals** which the function is searching for in the **relating_approval** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **approval_relationship** entity.

EXPRESS specification:

```
*)
END_SCHEMA; -- approval_schema
(*
```

4.6 contract_schema

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

EXPRESS specification:

```
*)
SCHEMA contract_schema;

REFERENCE FROM support_resource_schema
    (label,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.10, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.6.1 Introduction

This clause defines requirements for the **contract_schema**. Contracts are binding agreements.

4.6.2 Fundamental concepts and assumptions

Contract information can be attached to any aspect of a product data.

4.6.3 contract_schema entity definitions

4.6.3.1 contract_type

A **contract_type** is a sort of contract.

EXAMPLE 64 – A contract for an organization that is supplying goods to another organization could be "fixed" or "cost plus".

EXPRESS specification:

```
*)
ENTITY contract_type;
  description : label;
END_ENTITY;
(*
```

Attribute definitions:

description: a description of the sort of contract.

4.6.3.2 contract

A **contract** is a binding agreement.

NOTE – **Contracts** may be enforceable by law.

EXPRESS specification:

```
*)
ENTITY contract;
  name    : label;
  purpose : text;
  kind    : contract_type;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **contract** is referred to.

purpose: an informal description of the reason for the **contract**.

kind: the **contract**'s type.

EXPRESS specification:

```
*)
END_SCHEMA; -- contract_schema
(*
```

4.7 security_classification_schema

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

EXPRESS specification:

```
*)
SCHEMA security_classification_schema;

REFERENCE FROM support_resource_schema
    (label,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.11, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.7.1 Introduction

This clause defines requirements for the **security_classification_schema**. A security classification is the level of confidentiality that is required for the purpose of product data protection. Security classifications are assigned by some authoritative agency.

4.7.2 Fundamental concepts and assumptions

Security classification information can be attached to any aspect of product data.

4.7.3 security_classification_schema entity definitions

4.7.3.1 security_classification_level

A **security_classification_level** is a category of security.

EXAMPLE 65 – "Confidential", "secret", and "top secret" are examples of **security_classification_levels**.

EXPRESS specification:

```
*)
ENTITY security_classification_level;
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **security_classification_level** is referred to.

4.7.3.2 security_classification

A security classification is the level of confidentiality that is required for the purpose of product data protection.

Application protocols that use this resource construct shall specify the extent of the security classification.

EXPRESS specification:

```
*)
ENTITY security_classification;
  name           : label;
  purpose        : text;
  security_level : security_classification_level;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **security classification** is referred to.

purpose: an informal description of the intent of the **security_classification**.

security_level: the category of the **security_classification**.

EXPRESS specification:

```
*)
END_SCHEMA; -- security_classification_schema
(*
```

4.8 person_organization_schema

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

EXPRESS specification:

```
*)
SCHEMA person_organization_schema;

REFERENCE FROM support_resource_schema
    (bag_to_set,
     identifier,
     label,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.12, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.8.1 Introduction

This clause defines requirements for the **person_organization_schema**. People and organizations are associated with product data to provide data concerning administrative structures and individuals who serve as points of contact or have particular responsibility in those structures.

4.8.2 Fundamental concepts and assumptions

People and organizations may be associated with any aspect of product data.

4.8.3 person_organization_schema type definition: person_organization_select

The **person_organization_select** type allows a **person** and/or **organization** to be referenced.

EXPRESS specification:

```
*)
TYPE person_organization_select = SELECT
  (person,
   organization,
   person_and_organization);
END_TYPE;
(*
```

4.8.4 person_organization_schema entity definitions

4.8.4.1 address

An **address** is the place where people and organizations are located.

EXPRESS specification:

```
*)
ENTITY address;
  internal_location       : OPTIONAL label;
  street_number           : OPTIONAL label;
  street                  : OPTIONAL label;
  postal_box              : OPTIONAL label;
  town                    : OPTIONAL label;
  region                  : OPTIONAL label;
  postal_code             : OPTIONAL label;
  country                 : OPTIONAL label;
  facsimile_number        : OPTIONAL label;
  telephone_number        : OPTIONAL label;
  electronic_mail_address : OPTIONAL label;
  telex_number            : OPTIONAL label;
WHERE
  WR1: EXISTS(internal_location)       OR
       EXISTS(street_number)           OR
       EXISTS(street)                  OR
       EXISTS(postal_box)              OR
       EXISTS(town)                    OR
       EXISTS(region)                  OR
       EXISTS(postal_code)             OR
       EXISTS(country)                 OR
       EXISTS(facsimile_number)        OR
       EXISTS(telephone_number)        OR
       EXISTS(electronic_mail_address) OR
       EXISTS(telex_number);
END_ENTITY;
(*
```

Attribute definitions:

internal_location: an organization-defined address for internal mail delivery.

street_number: the number of a building in a street.

street: the name of a street.

postal_box: the number of a postal box.

town: the name of a town.

region: the name of a region.

EXAMPLE 66 – The counties of Great Britain and the states of the United States of America are examples of regions.

postal_code: the code that is used by the **country**'s postal service.

country: the name of a country.

facsimile_number: the number at which facsimiles may be received.

telephone_number: the number at which telephone calls may be received.

electronic_mail_address: the electronic address at which electronic mail may be received.

telex_number: the number at which telex messages may be received.

Formal propositions:

WR1: at least one of the attributes shall have a value.

4.8.4.2 personal_address

A **personal_address** is an **address** where people reside.

EXPRESS specification:

```
*)
ENTITY personal_address
  SUBTYPE OF (address);
  people      : SET [1:?] OF person;
  description : text;
END_ENTITY;
(*
```

Attribute definitions:

people: the people who reside at the **address**.

description: text that relates the nature of the **personal_address**.

4.8.4.3 organizational_address

An **organizational_address** is an **address** where **organization**s are located.

EXPRESS specification:

```
*)
ENTITY organizational_address
  SUBTYPE OF (address);
  organizations : SET [1:?] OF organization;
  description   : text;
END_ENTITY;
(*
```

Attribute definitions:

organizations: the **organization**s located at the **address**.

description: text that relates the nature of the **organizational_address**.

4.8.4.4 person

A **person** is an individual human being.

EXPRESS specification:

```
*)
ENTITY person;
  id            : identifier;
  last_name     : OPTIONAL label;
  first_name    : OPTIONAL label;
  middle_names  : OPTIONAL LIST [1:?] OF label;
  prefix_titles : OPTIONAL LIST [1:?] OF label;
  suffix_titles : OPTIONAL LIST [1:?] OF label;
UNIQUE
  UR1: id;
WHERE
  WR1: EXISTS(last_name) OR EXISTS(first_name);
END_ENTITY;
(*
```

Attribute definitions:

id: a means by which the **person** may be identified.

EXAMPLE 67 – In the USA a person's **id** would be his or her social security number whereas in Great Britain it would be his or her national insurance number.

last_name: the **person**'s surname.

first_name: the first element of the **person**'s list of forenames.

middle_names: the **person**'s other forenames, if there are any.

prefix_titles: the word, or group of words, which specify the **person**'s social and/or professional standing and appear before his or her names.

suffix_titles: the word, or group of words, which specify the **person**'s social and/or professional standing and appear after his or her names.

Formal propositions:

UR1: a **person**'s **id** shall be unique.

WR1: either the **last_name** or the **first_name** shall be defined.

4.8.4.5 organization

An **organization** is an administrative structure.

EXPRESS specification:

```
*)
ENTITY organization;
  id          : OPTIONAL identifier;
  name        : label;
```

```
  description : text;
END_ENTITY;
(*
```

Attribute definitions:

id: the means by which the **organization**'s individuality may be deduced.

name: the word or group of words by which the **organization** is referred to.

description: text that relates the nature of the **organization**.

4.8.4.6 organizational_project

An **organizational_project** is a project for which one or more **organizations** is responsible.

EXPRESS specification:

```
*)
ENTITY organizational_project;
  name                       : label;
  description                : text;
  responsible_organizations : SET[1:?] OF organization;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **organizational_project** is referred to.

description: text that relates the nature of the **organizational_project**.

responsible_organizations: the **organizations** which are responsible for the project.

4.8.4.7 person_and_organization

A **person_and_organization** is a person in an organization.

EXPRESS specification:

```
*)
ENTITY person_and_organization;
  the_person       : person;
  the_organization : organization;
END_ENTITY;
(*
```

Attribute definitions:

the_person: the **person** who is related to the **organization**.

the_organization: the **organization** to which the **person** is related.

4.8.4.8 organization_relationship

An **organization_relationship** establishes an association between two **organizations**.

EXAMPLE 68 – Each department of an enterprise, and the enterprise itself, may be regarded as individual organizations. The fact that the departments are a part of the whole enterprise can be captured using this entity.

The meaning of the relationship in a particular application context is defined in specializations of this entity.

NOTES

1 – Relationships captured using this entity may be parent-child relationships. Specializations of this entity state this fact if it is true for the particular specialization.

2 – This entity, in conjunction with the **organization** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY organization_relationship;
  name                  : label;
  description           : text;
  relating_organization : organization;
  related_organization  : organization;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **organization_relationship** is referred to.

description: text that relates the nature of the **organization_relationship**.

relating_organization: one of the **organizations** which is a part of the association.

related_organization: the other **organization** which is a part of the association. If one element of the relationship is dependent upon the other, this attribute shall be the dependent one.

4.8.4.9 person_and_organization_role

A **person_and_organization_role** is a role of a **person** in the context of an **organization**.

EXAMPLE 69 – The role of a **person** could be "buyer" in the context of the **organization** where he/she works and it could be "customer" in the context of the **organization** from which he/she purchases goods.

EXPRESS specification:

```
*)
ENTITY person_and_organization_role;
```

```
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the function being performed is referred to.

4.8.4.10 person_role

A **person_role** is a function which is performed by a **person**.

EXPRESS specification:

```
*)
ENTITY person_role;
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the function being performed is referred to.

4.8.4.11 organization_role

An **organization_role** is a function which is performed by an **organization**.

EXPRESS specification:

```
*)
ENTITY organization_role;
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the function being performed is referred to.

4.8.5 person_organization_schema function definition: acyclic_organization_relationship

The **acyclic_organization_relationship** function determines whether or not the given **organizations** have been self-defined by the associations made in the specified **organization_relationship**. This function may be used to evaluate either a **organization_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **organization_relationship** entity is either an **organization_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **organization_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_organization_relationship
         (relation          : organization_relationship;
          relatives         : SET OF organization;
          specific_relation : STRING) : LOGICAL;
  LOCAL
    x               : SET OF organization_relationship;
    i               : INTEGER;
    local_relatives : SET OF organization;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_organization :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_organization, specific_relation));
  local_relatives := relatives + relation.relating_organization;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_organization_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **organization_relationship** to be checked.

relatives: (input) the set of **organizations** which the function is searching for in the **relating_organization** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **organization_relationship** entity.

EXPRESS specification:

```
*)
END_SCHEMA; -- person_organization_schema
(*
```

4.9 date_time_schema

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

EXPRESS specification:

```
*)
SCHEMA date_time_schema;

REFERENCE FROM support_resource_schema
    (label);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.13, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.9.1 Introduction

This clause defines requirements for the **date_time_schema**. This schema allows representations of dates, both calendar and ordinal, time of day, combinations of date and time of day, and periods of time to be defined.

4.9.2 Fundamental concepts and assumptions

Any kind of product data may have a date and/or time associated with it.

4.9.3 date_time_schema type definitions

4.9.3.1 date_time_select

A **date_time_select** type allows a **date** and/or **local_time** to be referenced.

EXPRESS specification:

```
*)
TYPE date_time_select = SELECT
  (date,
   local_time,
   date_and_time);
END_TYPE;
(*
```

4.9.3.2 year_number

A **year_number** is the year as defined in the Gregorian Calendar.

NOTE – ISO 8601 defines the Gregorian Calendar.

EXPRESS specification:

```
*)
TYPE year_number = INTEGER;
END_TYPE;
(*
```

4.9.3.3 month_in_year_number

A **month_in_year_number** is the position of the specified month in a year as defined in ISO 8601 (subclause 5.2.1).

NOTE – January is month number 1, February is month number 2, March is month number 3, April is month number 4, May is month number 5, June is month number 6, July is month number 7, August is month number 8, September is month number 9, October is month number 10, November is month number 11, and December is month number 12.

EXPRESS specification:

```
*)
TYPE month_in_year_number = INTEGER;
WHERE
  WR1: { 1 <= SELF <= 12 };
END_TYPE;
(*
```

Formal propositions:

WR1: the value of the integer shall be between 1 and 12.

4.9.3.4 week_in_year_number

A **week_in_year_number** is the value of the calendar week as defined in ISO 8601 (subclause 3.1.7).

NOTE – Week number 1 is the week containing the first Thursday of the year. This is equivalent to saying that week number 1 contains the date 4th January. Assuming that 1st January, Saturdays and Sundays are not business days, this is equivalent to saying that the first business day of the year occurs in week 1.

EXPRESS specification:

```
*)
TYPE week_in_year_number = INTEGER;
WHERE
  WR1: { 1 <= SELF <= 53 };
END_TYPE;
(*
```

Formal propositions:

WR1: the value of the integer shall be between 1 and 53.

4.9.3.5 day_in_week_number

A **day_in_week_number** is the value of day as defined in ISO 8601 (subclause 5.2.3).

NOTE – Monday is day number 1, Tuesday is day number 2, Wednesday is day number 3, Thursday is day number 4, Friday is day number 5, Saturday is day number 6, and Sunday is day number 7.

EXPRESS specification:

```
*)
TYPE day_in_week_number = INTEGER;
WHERE
  WR1: { 1 <= SELF <= 7 };
END_TYPE;
(*
```

Formal propositions:

WR1: the value of the integer shall be between 1 and 7.

4.9.3.6 day_in_month_number

A **day_in_month_number** is the position of the specified day in a month.

EXPRESS specification:

```
*)
TYPE day_in_month_number = INTEGER;
END_TYPE;
(*
```

4.9.3.7 day_in_year_number

A **day_in_year_number** is the position of the specified day in a year.

EXAMPLE 70 – The 27th day of March is day 86 in years that are not leap years and day 87 in leap years.

EXPRESS specification:

```
*)
TYPE day_in_year_number = INTEGER;
END_TYPE;
(*
```

4.9.3.8 ahead_or_behind

The **ahead_or_behind** type is used to specify whether a given time is ahead of or behind coordinated universal time.

EXPRESS specification:

```
*)
TYPE ahead_or_behind = ENUMERATION OF
  (ahead,
   behind);
END_TYPE;
(*
```

4.9.3.9 hour_in_day

An **hour_in_day** is the hour element of a specified time on a 24 hour clock.

EXAMPLE 71 – The hour_in_day of 3 o'clock in the afternoon is 15.

NOTE – Although ISO 8061 allows two representations for midnight, 0000 and 2400, this Part of ISO 10303 restricts the representation to the first value.

EXPRESS specification:

```
*)
TYPE hour_in_day = INTEGER;
WHERE
  WR1: { 0 <= SELF < 24 };
END_TYPE;
(*
```

Formal propositions:

WR1: the value of the integer shall be between 0 and 23.

4.9.3.10 minute_in_hour

A **minute_in_hour** is the minute element of a specified time.

EXPRESS specification:

```
*)
TYPE minute_in_hour = INTEGER;
WHERE
  WR1: { 0 <= SELF <= 59 };
END_TYPE;
(*
```

Formal propositions:

WR1: the value of the integer shall be between 0 and 59.

4.9.3.11 second_in_minute

A **second_in_minute** is the second element of a specified time.

EXPRESS specification:

```
*)
TYPE second_in_minute = REAL;
WHERE
  WR1: { 0 <= SELF < 60 };
END_TYPE;
(*
```

Formal propositions:

WR1: the value of the real number shall be between 0 to 59.

4.9.4 date_time_schema entity definitions

4.9.4.1 date

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

EXPRESS specification:

```
*)
ENTITY date
  SUPERTYPE OF (ONEOF (calendar_date,
                       ordinal_date,
                       week_of_year_and_day_date));
  year_component : year_number;
END_ENTITY;
(*
```

Attribute definitions:

year_component: the year in which the **date** occurs.

4.9.4.2 calendar_date

A **calendar_date** is a **date** which is identified by a day in a month of a year.

EXPRESS specification:

```
*)
ENTITY calendar_date
  SUBTYPE OF (date);
  day_component   : day_in_month_number;
  month_component : month_in_year_number;
WHERE
  WR1: valid_calendar_date (SELF);
END_ENTITY;
(*
```

Attribute definitions:

day_component: the day element of the **date**.

month_component: the month element of the **date**.

Formal propositions:

WR1: when the **month_component** is "April", "June", "August", or "November" the **day_component** shall be between 1 and 30; when the **month_component** is "February" and the **year_component** is a leap year the **day_component** shall be between 1 and 29; when the **month_component** is "February" and the **year_component** is not a leap year the **day_component** shall be between 1 and 28; otherwise the **day_component** shall be between 1 and 31.

4.9.4.3 ordinal_date

An **ordinal_date** is a **date** which is identified by a day of a year.

EXPRESS specification:

```
*)
ENTITY ordinal_date
  SUBTYPE OF (date);
  day_component : day_in_year_number;
WHERE
  WR1: (NOT leap_year(SELF.year_component) AND { 1 <= day_component <= 365 }) OR
          (leap_year(SELF.year_component) AND { 1 <= day_component <= 366 });
END_ENTITY;
(*
```

day_component: the day element of the **date**.

Formal propositions:

WR1: the **day_component** shall be between 1 and 365 if the **year_component** is not a leap year; otherwise the **day_component** shall be between 1 and 366.

4.9.4.4 week_of_year_and_day_date

A **week_of_year_and_date** is a **date** which is identified by a day in a week of a year.

EXPRESS specification:

```
*)
ENTITY week_of_year_and_day_date
  SUBTYPE OF (date);
  week_component : week_in_year_number;
  day_component  : OPTIONAL day_in_week_number;
END_ENTITY;
(*
```

Attribute definitions:

week_component: the week element of the **date**.

day_component: the day element of the **date**.

Informal propositions:

valid_year_and_day: the combination of the **day_component** and the **week_component** shall be between 1 and 365 if the **year_component** is not a leap year, otherwise the combination of the **day_component** and the **week_component** shall be between 1 and 366.

4.9.4.5 coordinated_universal_time_offset

A **coordinated_universal_time_offset** is used to relate a time to coordinated universal time by an offset (specified in hours and minutes) and a direction.

EXPRESS specification:

```
*)
ENTITY coordinated_universal_time_offset;
  hour_offset   : hour_in_day;
  minute_offset : OPTIONAL minute_in_hour;
  sense         : ahead_or_behind;
END_ENTITY;
(*
```

Attribute definitions:

hour_offset: the number of hours by which a time is offset from coordinated universal time.

minute_offset: the number of minutes by which a time is offset from coordinated universal time.

sense: the direction of the offset.

4.9.4.6 local_time

A **local_time** is a moment of occurrence measured by hour, minute, and second. It represents one instant of time on a 24 hour clock.

NOTE – This construct is used to represent a moment of time whereas time measures (see clause 4.14) represent amounts of time.

EXAMPLE 72 – 1500 hours is an instant in time whereas 15 hours is an amount of time.

EXPRESS specification:

```
*)
ENTITY local_time;
  hour_component   : hour_in_day;
  minute_component : OPTIONAL minute_in_hour;
  second_component : OPTIONAL second_in_minute;
  zone             : coordinated_universal_time_offset;
WHERE
  WR1: valid_time (SELF);
END_ENTITY;
(*
```

Attribute definitions:

hour_component: the number of hours.

minute_component: the number of minutes.

second_component: the number of seconds.

zone: the relationship of the time to coordinated universal time.

Formal propositions:

WR1: the **seconds** attribute shall only exist if the **minute** attribute exists.

4.9.4.7 date_and_time

A **date_and_time** is a moment of time on a particular day.

EXPRESS specification:

```
*)
ENTITY date_and_time;
  date_component : date;
  time_component : local_time;
END_ENTITY;
(*
```

Attribute definitions:

date_component: the date element of the date time combination.

time_component: the time element of the date time combination.

4.9.4.8 date_time_role

A **date_time_role** is an event that is being date and time stamped. Dates and times are associated with product data to indicate when a particular event happened.

EXAMPLE 73 – The role of a date and time could be "completed on" or "due by".

EXPRESS specification:

```
*)
ENTITY date_time_role;
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the event that the combination of the date and time captures is referred to.

4.9.4.9 date_role

A **date_role** is an event that is being date stamped. Dates are associated with product data to indicate when a particular event happened.

EXPRESS specification:

```
*)
ENTITY date_role;
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the event that the date captures is referred to.

4.9.4.10 time_role

A **time_role** is an event that is being time stamped. Times are associated with product data to indicate when a particular event happened.

EXPRESS specification:

```
*)
ENTITY time_role;
  name : label;
END_ENTITY;
```

```
(*
```

Attribute definitions:

name: the word or group of words by which the event that the time captures is referred to.

4.9.5 date_time_schema function definitions

4.9.5.1 leap_year

The **leap_year** function determines whether a given year is a leap year or not according to the Gregorian calendar algorithm. It returns TRUE if the year is a leap year, otherwise it returns FALSE.

EXPRESS specification:

```
*)
FUNCTION leap_year(year : year_number) : BOOLEAN;

  IF ((((year MOD 4) = 0) AND ((year MOD 100) <> 0)) OR
       ((year MOD 400) = 0)) THEN
    RETURN(TRUE);
  ELSE
    RETURN(FALSE);
  END_IF;
END_FUNCTION;
(*
```

Argument definitions:

year: (input) the candidate **year_number** that is being checked.

4.9.5.2 valid_calendar_date

The **valid_calendar_date** function determines whether the components of a **calendar_date** indicate a valid **date**. If the **calendar_date** is valid, the function returns TRUE, otherwise it returns FALSE.

EXPRESS specification:

```
*)
FUNCTION valid_calendar_date (date : calendar_date) : LOGICAL;

  IF NOT ({1 <= date.day_component <= 31}) THEN
    RETURN(FALSE);
  END_IF;
```

```
  CASE date.month_component OF
    4          : RETURN({ 1<= date.day_component <= 30});
    6          : RETURN({ 1<= date.day_component <= 30});
    9          : RETURN({ 1<= date.day_component <= 30});
    11         : RETURN({ 1<= date.day_component <= 30});
    2          :
    BEGIN
      IF (leap_year(date.year_component)) THEN
        RETURN({ 1<= date.day_component <= 29});
      ELSE
        RETURN({ 1<= date.day_component <= 28});
      END_IF;
    END;
    OTHERWISE : RETURN(TRUE);
  END_CASE;
END_FUNCTION;
(*
```

Argument definitions:

date: (input) the candidate **calendar_date** that is to be checked.

4.9.5.3 valid_time

The **valid_time** function determines whether a candidate **local_time** has a **minute_component** if is has a **second_component**. It returns FALSE if the condition is not met, otherwise it returns TRUE.

EXPRESS specification:

```
*)
FUNCTION valid_time (time: local_time) : BOOLEAN;
  IF EXISTS (time.second_component) THEN
    RETURN (EXISTS (time.minute_component));
  ELSE
    RETURN (TRUE);
  END_IF;
END_FUNCTION;
(*
```

Argument definitions:

time: (input) the candidate **local_time** that is to be checked.

EXPRESS specification:

```
*)
```

```
END_SCHEMA; -- date_time_schema
(*
```

4.10 group_schema

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

EXPRESS specification:

```
*)
SCHEMA group_schema;

REFERENCE FROM support_resource_schema
    (label,
     bag_to_set,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.14, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.10.1 Introduction

This clause defines requirements for the **group_schema**. Groups are collections of product data which have a common identity that is defined by human beings when they create product data. The structure specified in this schema allows the common identity to be captured; it does not specify, or allow the specification of, the meaning of the common identity or the criteria for the common identity. The only semantics is a name; no other semantics exist.

NOTE – It is recognised that common industrial practice implies other semantics. These implications are not supported by the resource constructs in this schema.

EXAMPLE 74 – A group may contain descriptions of a ship, cars, and bicycles. The reason for the grouping could be that they are all vehicles or it could be that the cars and bicycles are to be transported in the ship. The fact that the grouping exists can be specified with the resource constructs defined in this schema but the meaning of the grouping cannot be specified.

4.10.2 Fundamental concepts and assumptions

Any kinds of product data may be grouped together.

4.10.3 group_schema entity definitions

4.10.3.1 group

A **group** is an identification of a collection of elements.

NOTE 1 – This entity, in conjunction with the **group_relationship** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY group;
  name        : label;
  description : text;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **group** is referred to.

description: text that relates the nature of the **group**.

4.10.3.2 group_relationship

A **group_relationship** is an association between **groups**.

NOTES

1 – Relationships captured using this entity may be parent-child relationships. Specializations of this entity state this fact if it is true for the particular specialization.

2 – This entity, in conjunction with the **group** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY group_relationship;
  name           : label;
  description    : text;
  relating_group : group;
  related_group  : group;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **group_relationship** is referred to.

description: text that relates the nature of the **group_relationship**.

relating_group: one of the **group** entities which is a part of the relationship.

NOTE 3 – The role of this attribute is defined in the application protocol or the ISO 10303 integrated resource that uses or specializes this entity.

related_group: the other **group** entity which is a part of the relationship. If one element of the relationship is dependent upon the other, this attribute shall be the dependent one.

NOTE 4 – The role of this attribute is defined in the application protocol or the ISO 10303 integrated resource that uses or specializes this entity.

4.10.4 group_schema function definition: acyclic_group_relationship

The **acyclic_group_relationship** function determines whether or not the given **groups** have been self-defined by the associations made in the specified **group_relationship**. This function may be used to evaluate either a **group_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **group_relationship** entity is either a **group_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **group_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_group_relationship
  (relation          : group_relationship;
   relatives         : SET OF group;
   specific_relation : STRING) : LOGICAL;

  LOCAL
    x               : SET OF group_relationship;
    i               : INTEGER;
    local_relatives : SET OF group;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_group :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN (relation.relating_group, specific_relation));
  local_relatives := relatives + relation.relating_group;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_group_relationship
```

```
            (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **group_relationship** to be checked.

relatives: (input) the set of **groups** which the function is searching for in the **relating_group** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **group_relationship** entity.

EXPRESS specification:

```
*)
END_SCHEMA; -- group_schema
(*
```

4.11 effectivity_schema

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

EXPRESS specification:

```
*)
SCHEMA effectivity_schema;

  REFERENCE FROM measure_schema
    (measure_with_unit);

  REFERENCE FROM support_resource_schema
    (identifier);

  REFERENCE FROM date_time_schema
    (date_and_time);

(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

measure_schema	clause 4.14 of this part of ISO 10303
support_resource_schema	clause 4.13 of this part of ISO 10303
date_time_schema	clause 4.9 of this part of ISO 10303

2 – See annex F, figure F.15, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.11.1 Introduction

This clause defines requirements for the **effectivity_schema**. The subject of the **effectivity_schema** is the relationship between aspects of product data and the life cycle management tracking that is to be maintained against those aspects of product data over time.

This schema supports the representation of effectivity according to the following criteria:

- from the identification of a given batch of aspects of product data;
- from the identification of a given aspect of product data by serial number;
- aspects of product data realized from a given date.

4.11.2 Fundamental concepts and assumptions

Effectivity information can be attached to any aspect of product data.

An organization determines which aspects of product data are to be under its configuration control. These aspects of product data become the configuration items of the organization. These are high level functional elements which act as the focal points for managing the effectivity of other, related, items.

Only effectivities based on date, serial number and lot number are required for this edition of this part of ISO 10303.

4.11.3 effectivity_schema entity definitions

4.11.3.1 effectivity

An **effectivity** is the identification of the valid use of an aspect of product data. An **effectivity** may be tracked by date, serial number, or lot number.

Application protocols that use this resource construct shall specify the aspect of product data that requires the effectivity concept and the method of which effectivity is tracked.

EXPRESS specification:

```
*)
ENTITY effectivity
SUPERTYPE OF (ONEOF (serial_numbered_effectivity,
                     dated_effectivity,
                     lot_effectivity));
  id     : identifier;
END_ENTITY;
(*
```

Attribute definitions:

id: an identification of the effectivity.

4.11.3.2 serial_numbered_effectivity

A **serial_numbered_effectivity** is an **effectivity** that applies from a given serial number.

EXPRESS specification:

```
*)
ENTITY serial_numbered_effectivity
   SUBTYPE OF (effectivity);
   effectivity_start_id :  identifier;
   effectivity_end_id   :  OPTIONAL identifier;
END_ENTITY;
(*
```

Attribute definitions:

effectivity_start_id: the serial number of the first aspect of product data that the **effectivity** applies to.

effectivity_end_id: the serial number of the last aspect of product data that the **effectivity** applies to. If a value for this attribute is not defined then the **effectivity** has no defined end. If an effectivity is for a single aspect of product data then the values of both attributes shall be equal.

4.11.3.3 dated_effectivity

A **dated_effectivity** is an **effectivity** that applies from a given date and/or time.

EXPRESS specification:

```
*)
ENTITY dated_effectivity
   SUBTYPE OF (effectivity);
   effectivity_start_date : date_and_time;
   effectivity_end_date   : OPTIONAL date_and_time;
END_ENTITY;
(*
```

Attribute definitions:

effectivity_start_date: the date and/or time when the **effectivity** starts.

effectivity_end_date: the date and/or time when the **effectivity** ends. If a value for this attribute is not defined then the **effectivity** has no defined end.

4.11.3.4 lot_effectivity

A **lot_effectivity** is an **effectivity** that applies from a given batch of aspects of product data.

EXPRESS specification:

```
*)
ENTITY lot_effectivity
   SUBTYPE OF (effectivity);
   effectivity_lot_id    : identifier;
   effectivity_lot_size  : measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

effectivity_lot_id: the identification of the batch of aspects of product data that the **effectivity** applies to.

effectivity_lot_size: the size of the batch of aspects of product data that the **effectivity** applies to.

EXPRESS specification:

```
*)
END_SCHEMA; -- effectivity_schema
(*
```

4.12 external_reference_schema

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

EXPRESS specification:

```
*)
SCHEMA external_reference_schema;

REFERENCE FROM support_resource_schema
    (label,
     text,
```

```
    identifier,
    bag_to_set);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema clause 4.13 of this part of ISO 10303

2 – See annex F, figure F.16, for a graphical presentation of this schema.

3 – This schema contains support resources.

4.12.1 Introduction

This clause defines requirements for the **external_reference_schema**. This schema provides a means of identifying information that is not explicitly represented in a given exchange; the identified information is defined in either the application protocol to which the exchange conforms or an identified external source. Information may be supplied to an external system for use in the generation of the information that is not explicitly represented in the given exchange.

4.12.2 Fundamental concepts and assumptions

There is a requirement to be able to refer to information that is not explicitly represented in a given exchange. This information is either predefined, in the application protocol to which the exchange conforms, or is defined elsewhere. When the information is defined in the application protocol to which the exchange conforms, this requirement is satisfied by a reference which identifies the relevant information in the application protocol. Otherwise, this requirement is satisfied by a reference which identifies the relevant information and its source.

4.12.3 external_reference_schema type definitions

4.12.3.1 message

A **message** is a communication which is addressed to a system in order to trigger some action. The result of such an action is an **externally_defined_item**.

NOTE – The legal values for the message are specified within an application interpreted model.

EXPRESS specification:

```
*)
TYPE message = STRING;
END_TYPE;
(*
```

4.12.3.2 reference

A **reference** is a means of identifying and retrieving an **externally_defined_item**.

EXPRESS specification:

```
*)
TYPE source_item = SELECT (identifier, message);
END_TYPE;
(*
```

4.12.4 external_reference_schema entity definitions

4.12.4.1 external_source

An **external_source** is the identification of a source of product data that is not the application protocol to which the exchange conforms.

NOTE – The product data may conform to some other part of this International Standard.

EXPRESS specification:

```
*)
ENTITY external_source;
  source_id : source_item;
END_ENTITY;
(*
```

Attribute definitions:

source_id: the identification of the **external_source**.

4.12.4.2 external_source_relationship

An **external_source_relationship** is an association between two **external_sources**.

NOTES

1 – One **external_source** may be a subset of another **external_source**. Specializations of this entity state this fact if it is true for the particular specialization.

2 – This entity, in conjunction with the **external_source** entity, is based on the relationship template that is described in annex D.

EXPRESS specification:

```
*)
ENTITY external_source_relationship;
  name           : label;
  description    : text;
```

```
  relating_source : external_source;
  related_source  : external_source;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **external_source_relationship** is referred to.

description: text that relates the nature of the **external_source_relationship**.

relating_source: one of the **external_sources** which is a part of the relationship.

related_source: the other **external_source** which is a part of the relationship. If one element of the relationship is dependent upon the other then this attribute shall be the dependent one.

4.12.4.3 pre_defined_item

A **pre_defined_item** is the identification of information that is not explicitly represented in a given exchange but that is defined in the application protocol to which the exchange conforms.

EXAMPLE 75 – A reference to the colour "red" without any definition of the associated red-green-blue values would be a **pre_defined_item** if the red-green-blue values of the colour "red" were specified in the relevant application protocol.

EXPRESS specification:

```
*)
ENTITY pre_defined_item;
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **pre_defined_item** is referred to.

4.12.4.4 externally_defined_item

An **externally_defined_item** is the identification of information that is not explicitly represented in a given exchange and that is not defined in the application protocol to which the exchange conforms.

NOTE – Unlike **pre_defined_items**, support for the complete definition of an **externally_defined_item** is not a conformance requirement for implementations which support application protocols that use this entity.

EXPRESS specification:

```
*)
ENTITY externally_defined_item;
  item_id : source_item;
  source  : external_source;
END_ENTITY;
(*
```

Attribute definitions:

item_id: the identification of the referent item.

source: an **external_source** which contains the referent item.

4.12.5 external_reference_schema function definition: acyclic_external_source_relationship

The **acyclic_external_source_relationship** function determines whether or not the given **external_sources** have been self-defined by the associations made in the specified **external_source_relationship**. This function may be used to evaluate either a **external_source_relationship** or any of its subtypes.

NOTE 1 – A specified type of the **external_source_relationship** entity is either an **external_source_relationship** or one of its subtypes.

The function returns a value of TRUE if none of the elements of the **relatives** argument occur in the **relation** argument of the type which is given in the **specific_relation** argument. Otherwise it returns a value of FALSE.

NOTE 2 – This function is not used in this schema. It is defined here because other ISO 10303 integrated resources and application protocols that use the **external_source_relationship** entity include rules that use this function.

EXPRESS specification:

```
*)
FUNCTION acyclic_external_source_relationship
  (relation          : external_source_relationship;
   relatives         : SET OF external_source;
   specific_relation : STRING) : LOGICAL;

  LOCAL
    x               : SET OF external_source_relationship;
    i               : INTEGER;
    local_relatives : SET OF external_source;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_source :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
```

```
  x := bag_to_set(USEDIN (relation.relating_source, specific_relation));
  local_relatives := relatives + relation.relating_source;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_external_source_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: (input) the candidate **external_source_relationship** to be checked.

relatives: (input) the set of **external_sources** which the function is searching for in the **relating_external_source** parameter of the **relation** argument.

specific_relation: (input) the fully qualified name of a subtype of the **external_source_relationship** entity.

4.12.6 End of schema declaration

```
*)
END_SCHEMA; -- external_reference_schema
(*
```

4.13 support_resource_schema

The following EXPRESS declaration begins the **support_resource_schema.**

EXPRESS specification:

```
*)
SCHEMA support_resource_schema;
(*
```

NOTE 1 – See annex F, figure F.17, for a graphical presentation of this schema.

NOTE 2 – This schema contains support resources.

4.13.1 Introduction

This clause defines requirements for the **support_resource_schema**. This schema contains EXPRESS constructs that are shared by more than one of the ISO 10303 integrated resource schemas.

NOTE – In contrast to elements of the generic product description resource schemas, which are also potentially referenced by more than one ISO 10303 integrated resource schema, the elements of this schema cannot exist without (that is, are existent-dependent upon) the entities that reference them.

4.13.2 Fundamental concepts and assumptions

It is desirable that the ISO 10303 integrated resource schemas are consistent with each other. The use of a set of common shareable resources is one aspect of consistency. Shared resource constructs are only specified once.

4.13.3 support_resource_schema type definitions

4.13.3.1 identifier

An **identifier** is an alphanumeric string which allows an individual thing to be identified. It may not provide natural-language meaning.

EXAMPLE 76 – A part number would be an identifier.

EXPRESS specification:

```
*)
TYPE identifier = STRING;
END_TYPE;
(*
```

4.13.3.2 label

A **label** is the term by which something may be referred to. It is a string which represents the human-interpretable name of something and shall have a natural-language meaning.

EXAMPLE 77 – "Smith", "Widget Inc.", and "Materials Test Laboratory" are examples of labels.

When necessary, it is the task of an application protocol to prescribe allowable values for this type.

EXPRESS specification:

```
*)
TYPE label = STRING;
END_TYPE;
(*
```

4.13.3.3 text

A **text** is an alphanumeric string of characters which is intended to be read and understood by a human being. It is for information purposes only.

EXPRESS specification:

```
*)
TYPE text = STRING;
END_TYPE;
(*
```

4.13.4 support_resource_schema function definition: bag_to_set

This function converts BAGs into SETs.

EXAMPLE 78 – It can be used to convert the BAGs returned by the USEDIN function into SETs that can be properly assigned to variables that are SETs.

EXPRESS specification:

```
*)
FUNCTION bag_to_set (the_bag : BAG OF GENERIC : intype) : SET OF GENERIC : intype;

  LOCAL
    the_set: SET OF GENERIC : intype := [];
    i      : INTEGER;
  END_LOCAL;

  IF SIZEOF (the_bag) > 0 THEN
    REPEAT i := 1 to HIINDEX (the_bag);
      the_set := the_set + the_bag [i];
    END_REPEAT;
  END_IF;

  RETURN (the_set);

END_FUNCTION;
(*
```

Argument definitions:

the_bag: the BAG that is to be converted into a SET.

EXPRESS specification:

```
*)
END_SCHEMA; -- support_resource_schema
(*
```

4.14 measure_schema

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

EXPRESS specification:

```
*)
SCHEMA measure_schema;

REFERENCE FROM support_resource_schema
    (label);

REFERENCE FROM representation_schema
    (representation_context);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

support_resource_schema	clause 4.13 of this part of ISO 10303
representation_schema	ISO 10303-43

2 – See annex F, figures F.18, F.19, and F.20, for a graphical presentation of this schema.

3 – This schema contains support resources.

4 – Examples of the ways in which the resource constructs in this schema may be used are given in annex E.

4.14.1 Introduction

This clause defines requirements for the **measure_schema**. This schema contains resource constructs that allow physical quantities to be described.

EXAMPLE 79 – Density, length, force, and time are different kinds of physical quantities.

The resource constructs defined in this schema are based upon material in ISO 31 and ISO 1000.

4.14.2 Fundamental concepts and assumptions

The following requirements are supported by this schema:

a) It shall be possible to specify the specific kinds of physical quantity in the ISO 10303 integrated resource schemas when the kind of a physical quantity is known at the time when the schema is being specified;

EXAMPLE 80 – The fact that the x, y, and z components of a cartesian point are distances is well accepted. It shall be possible to state this fact in any schema where a cartesian point entity is specified.

b) It shall be possible to specify non-specific kinds of physical quantity in the ISO 10303 integrated resource schemas when the kind of a physical quantity is not known at the time when the schema is being specified.

EXAMPLE 81 – The elements of a list of material properties will be physical quantities. Each element may be a different kind of physical quantity. The kind of each element is only decided upon at instantiation time.

4.14.3 measure_schema type definitions

4.14.3.1 measure_value

A **measure_value** is a value as defined in ISO 31-0 (clause 2).

EXPRESS specification:

```
*)
TYPE measure_value = SELECT
  (length_measure,
   mass_measure,
   time_measure,
   electric_current_measure,
   thermodynamic_temperature_measure,
   amount_of_substance_measure,
   luminous_intensity_measure,
   plane_angle_measure,
   solid_angle_measure,
   area_measure,
   volume_measure,
   ratio_measure,
   parameter_value,
   numeric_measure,
   context_dependent_measure,
   descriptive_measure,
```

```
   positive_length_measure,
   positive_plane_angle_measure,
   positive_ratio_measure,
   count_measure);
END_TYPE;
(*
```

4.14.3.2 length_measure

A **length_measure** is the value of a distance.

EXPRESS specification:

```
*)
TYPE length_measure = REAL;
END_TYPE;
(*
```

4.14.3.3 mass_measure

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

EXPRESS specification:

```
*)
TYPE mass_measure = REAL;
END_TYPE;
(*
```

4.14.3.4 time_measure

A **time_measure** is the value of the duration of periods.

EXPRESS specification:

```
*)
TYPE time_measure = REAL;
END_TYPE;
(*
```

4.14.3.5 electric_current_measure

An **electric_current_measure** is the value for the movement of electrically charged particles.

EXPRESS specification:

```
*)
TYPE electric_current_measure = REAL;
END_TYPE;
(*
```

4.14.3.6 thermodynamic_temperature_measure

A **thermodynamic_temperature_measure** is the value for the degree of heat of a body.

EXPRESS specification:

```
*)
TYPE thermodynamic_temperature_measure = REAL;
END_TYPE;
(*
```

4.14.3.7 amount_of_substance_measure

An **amount_of_substance_measure** is the value for the quantity of a substance when compared with the number of atoms in 0.012kilogram of carbon 12.

EXPRESS specification:

```
*)
TYPE amount_of_substance_measure = REAL;
END_TYPE;
(*
```

4.14.3.8 luminous_intensity_measure

A **luminous_intensity_measure** is the value for the brightness of a body.

EXPRESS specification:

```
*)
TYPE luminous_intensity_measure = REAL;
END_TYPE;
(*
```

4.14.3.9 plane_angle_measure

A **plane_angle_measure** is the value of an angle in a plane.

EXPRESS specification:

```
*)
TYPE plane_angle_measure = REAL;
END_TYPE;
(*
```

4.14.3.10 solid_angle_measure

A **solid_angle_measure** is the value of a solid angle.

EXPRESS specification:

```
*)
TYPE solid_angle_measure = REAL;
END_TYPE;
(*
```

4.14.3.11 area_measure

An **area_measure** is the value of the extent of a surface.

EXPRESS specification:

```
*)
TYPE area_measure = REAL;
END_TYPE;
(*
```

4.14.3.12 volume_measure

A **volume_measure** is the value of the solid content of a body.

EXPRESS specification:

```
*)
TYPE volume_measure = REAL;
END_TYPE;
(*
```

4.14.3.13 ratio_measure

A **ratio_measure** is the value of the relation between two physical quantities that are of the same kind.

EXPRESS specification:

```
*)
TYPE ratio_measure = REAL;
END_TYPE;
(*
```

4.14.3.14 parameter_value

A **parameter_value** is the value which specifies the amount of a parameter in some parameter space.

EXPRESS specification:

```
*)
TYPE parameter_value = REAL;
END_TYPE;
(*
```

4.14.3.15 numeric_measure

A **numeric_measure** is the numeric value of a physical quantity.

EXPRESS specification:

```
*)
TYPE numeric_measure = NUMBER;
END_TYPE;
(*
```

4.14.3.16 positive_length_measure

A **positive_length_measure** is a **length_measure** that is greater than zero.

EXPRESS specification:

```
*)
TYPE positive_length_measure = length_measure;
WHERE
  WR1: SELF > 0;
END_TYPE;
(*
```

Formal propositions:

WR1: the value shall be positive.

4.14.3.17 positive_plane_angle_measure

A **positive_plane_angle_measure** is a **plane_angle_measure** that is greater than zero.

EXPRESS specification:

```
*)
TYPE positive_plane_angle_measure = plane_angle_measure;
WHERE
  WR1: SELF > 0;
END_TYPE;
(*
```

Formal propositions:

WR1: the value shall be positive.

4.14.3.18 positive_ratio_measure

A **positive_ratio_measure** is a **ratio_measure** that is greater than zero.

EXPRESS specification:

```
*)
TYPE positive_ratio_measure = ratio_measure;
WHERE
  WR1: SELF > 0;
END_TYPE;
(*
```

Formal propositions:

WR1: the value shall be positive.

4.14.3.19 context_dependent_measure

A **context_dependent_measure** is the value of a physical quantity as defined by an application context.

EXPRESS specification:

```
*)
TYPE context_dependent_measure = REAL;
END_TYPE;
(*
```

4.14.3.20 descriptive_measure

A **descriptive_measure** is a human-interpretable definition of a quantifiable value.

EXPRESS specification:

```
*)
TYPE descriptive_measure = STRING;
END_TYPE;
(*
```

4.14.3.21 count_measure

A **count_measure** is the value of a count.

EXPRESS specification:

```
*)
TYPE count_measure = NUMBER;
END_TYPE;
(*
```

4.14.3.22 unit

A **unit** is a physical quantity, with a value of one, which is used as a standard in terms of which other quantities are expressed.

EXPRESS specification:

```
*)
TYPE unit = SELECT
  (named_unit,
   derived_unit);
END_TYPE;
(*
```

4.14.3.23 si_unit_name

An **si_unit_name** is the name of an SI unit. The definitions of the names of SI units are specified in ISO 1000 (clause 2).

EXPRESS specification:

```
*)
TYPE si_unit_name = ENUMERATION OF
  (metre,
   gram,
   second,
   ampere,
   kelvin,
   mole,
   candela,
   radian,
   steradian,
   hertz,
   newton,
   pascal,
   joule,
   watt,
   coulomb,
   volt,
   farad,
   ohm,
   siemens,
   weber,
   tesla,
   henry,
   degree_Celsius,
   lumen,
   lux,
   becquerel,
   gray,
   sievert);
END_TYPE;
(*
```

Enumerated item definitions:

metre: see ISO 1000 (subclause 2.1).

gram: see ISO 1000 (subclause 2.1).

NOTE – ISO 1000 (subclause 2.1) gives "kilogram" as the SI unit name. This part of ISO 10303 uses "gram" as the SI unit name to avoid unnecessary complication in the structure of this schema.

second: see ISO 1000 (subclause 2.1).

ampere: see ISO 1000 (subclause 2.1).

kelvin: see ISO 1000 (subclause 2.1).

mole: see ISO 1000 (subclause 2.1).

candela: see ISO 1000 (subclause 2.1).

radian: see ISO 1000 (subclause 2.2).

steradian: see ISO 1000 (subclause 2.2).

hertz: see ISO 1000 (subclause 2.3).

newton: see ISO 1000 (subclause 2.3).

pascal: see ISO 1000 (subclause 2.3).

joule: see ISO 1000 (subclause 2.3).

watt: see ISO 1000 (subclause 2.3).

coulomb: see ISO 1000 (subclause 2.3).

volt: see ISO 1000 (subclause 2.3).

farad: see ISO 1000 (subclause 2.3).

ohm: see ISO 1000 (subclause 2.3).

siemens: see ISO 1000 (subclause 2.3).

weber: see ISO 1000 (subclause 2.3).

tesla: see ISO 1000 (subclause 2.3).

henry: see ISO 1000 (subclause 2.3).

degree_Celsius: see ISO 1000 (subclause 2.3).

lumen: see ISO 1000 (subclause 2.3).

lux: see ISO 1000 (subclause 2.3).

becquerel: see ISO 1000 (subclause 2.3).

gray: see ISO 1000 (subclause 2.3).

sievert: see ISO 1000 (subclause 2.3).

4.14.3.24 si_prefix

An **si_prefix** is the name of a prefix that may be associated with an **si_unit**. The definitions of SI prefixes are specified in ISO 1000 (clause 3).

EXPRESS specification:

```
*)
TYPE si_prefix = ENUMERATION OF
  (exa,
   peta,
   tera,
   giga,
   mega,
   kilo,
   hecto,
   deca,
   deci,
   centi,
   milli,
   micro,
   nano,
   pico,
   femto,
   atto);
END_TYPE;
(*
```

Enumerated item definitions:

exa: see ISO 1000 (clause 3).

peta: see ISO 1000 (clause 3).

tera: see ISO 1000 (clause 3).

giga: see ISO 1000 (clause 3).

mega: see ISO 1000 (clause 3).

kilo: see ISO 1000 (clause 3).

hecto: see ISO 1000 (clause 3).

deca: see ISO 1000 (clause 3).

deci: see ISO 1000 (clause 3).

centi: see ISO 1000 (clause 3).

milli: see ISO 1000 (clause 3).

micro: see ISO 1000 (clause 3).

nano: see ISO 1000 (clause 3).

pico: see ISO 1000 (clause 3).

femto: see ISO 1000 (clause 3).

atto: see ISO 1000 (clause 3).

4.14.4 measure_schema entity definitions

4.14.4.1 named_unit

A **named_unit** is a unit quantity associated with the word, or group of words, by which the unit is identified.

EXPRESS specification:

```
*)
ENTITY named_unit
  SUPERTYPE OF (ONEOF (si_unit, conversion_based_unit, context_dependent_unit)
                ANDOR
                ONEOF (length_unit,
                       mass_unit,
                       time_unit,
                       electric_current_unit,
                       thermodynamic_temperature_unit,
                       amount_of_substance_unit,
                       luminous_intensity_unit,
                       plane_angle_unit,
                       solid_angle_unit,
                       area_unit,
                       volume_unit,
                       ratio_unit));
  dimensions : dimensional_exponents;
END_ENTITY;
(*
```

Attribute definitions:

dimensions: the exponents of the base properties by which the **named_unit** is defined.

4.14.4.2 si_unit

An **si_unit** is the fixed quantity used as a standard in terms of which items are measured as defined by ISO 1000 (clause 2).

EXPRESS specification:

```
*)
ENTITY si_unit
  SUBTYPE OF (named_unit);
  prefix     : OPTIONAL si_prefix;
  name       : si_unit_name;
DERIVE
  SELF\named_unit.dimensions : dimensional_exponents
                             := dimensions_for_si_unit (SELF.name);
```

```
END_ENTITY;
(*
```

Attribute definitions:

prefix: the SI prefix.

name: the word or group of words by which the **si_unit** is referred to.

4.14.4.3 conversion_based_unit

A **conversion_based_unit** is a unit that is defined based on a **measure_with_unit**.

EXAMPLE 82 – An inch is a **converted_unit**. It is from the Imperial system, its name is "inch" and it can be related to the **si_unit**, millimetre, through a **measure_with_unit** whose value is 25.4 millimetre. A foot is also a **converted_unit**. It is from the Imperial system, its name is "foot" and it can be related to an **si_unit**, millimetre, either directly or through the unit called "inch".

EXPRESS specification:

```
*)
ENTITY conversion_based_unit
  SUBTYPE OF (named_unit);
  name              : label;
  conversion_factor : measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **conversion_based_unit** is referred to.

conversion_factor: the physical quantity from which the **converted_unit** is derived.

4.14.4.4 context_dependent_unit

An **context_dependent_unit** is a unit which is not related to the SI system.

EXAMPLE 83 – The number of parts in an assembly is a physical quantity measured in units that may be called "parts" but which cannot be related to an SI unit.

EXPRESS specification:

```
*)
ENTITY context_dependent_unit
  SUBTYPE OF (named_unit);
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word or group of words by which the **context_dependent_unit** is referred to.

4.14.4.5 length_unit

A **length_unit** is the unit in which distances are measured.

EXPRESS specification:

```
*)
ENTITY length_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 1.0)                    AND
       (SELF\named_unit.dimensions.mass_exponent = 0.0)                      AND
       (SELF\named_unit.dimensions.time_exponent = 0.0)                      AND
       (SELF\named_unit.dimensions.electric_current_exponent = 0.0)          AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0)       AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY;
(*
```

Formal propositions:

WR1: the dimensional exponent of length shall be equal to one and all the other dimensional exponents shall be equal to zero.

4.14.4.6 mass_unit

A **mass_unit** is the unit in which the amount of matter that a body contains is measured.

EXPRESS specification:

```
*)
ENTITY mass_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.mass_exponent                      = 1.0) AND
       (SELF\named_unit.dimensions.time_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.electric_current_exponent          = 0.0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent       = 0.0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent        = 0.0);
END_ENTITY ;
(*
```

Formal propositions:

WR1: the dimensional exponent of mass shall be equal to one and all the other dimensional exponents shall be equal to zero.

4.14.4.7 time_unit

A **time_unit** is the unit in which the duration of periods is measured.

EXPRESS specification:

```
*)
ENTITY time_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.mass_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.time_exponent                      = 1.0) AND
       (SELF\named_unit.dimensions.electric_current_exponent          = 0.0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent       = 0.0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent        = 0.0);
END_ENTITY;
(*
```

Formal propositions:

WR1: the dimensional exponent of time shall be equal to one and all the other dimensional exponents shall be equal to zero.

4.14.4.8 electric_current_unit

An **electric_current_unit** is the unit in which the movement of electrically charged particles is measured.

EXPRESS specification:

```
*)
ENTITY electric_current_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.mass_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.time_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.electric_current_exponent          = 1.0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent       = 0.0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent        = 0.0);
END_ENTITY;
(*
```

Formal propositions:

WR1: the dimensional exponent of electric current shall be equal to one and all the other

dimensional exponents shall be equal to zero.

4.14.4.9 thermodynamic_temperature_unit

A **thermodynamic_temperature_unit** is the unit in which the degree of heat of a body is measured.

EXPRESS specification:

```
*)
ENTITY thermodynamic_temperature_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.mass_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.time_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.electric_current_exponent          = 0.0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 1.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent       = 0.0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent        = 0.0);
END_ENTITY;
(*
```

Formal propositions:

WR1: the dimensional exponent of thermodynamic temperature shall be equal to one and all the other dimensional exponents shall be equal to zero.

4.14.4.10 amount_of_substance_unit

An **amount_of_substance_unit** is the unit in which the quantity of a substance when compared with the number of atoms in 0.012 kilograms of carbon 12 is measured.

EXPRESS specification:

```
*)
ENTITY amount_of_substance_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.mass_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.time_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.electric_current_exponent          = 0.0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent       = 1.0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent        = 0.0);
END_ENTITY;
(*
```

Formal propositions:

WR1: the dimensional exponent of amount of substance shall be equal to one and all the other dimensional exponents shall be equal to zero.

4.14.4.11 luminous_intensity_unit

A **luminous_intensity_unit** is the unit in which the brightness of a body is measured.

EXPRESS specification:

```
*)
ENTITY luminous_intensity_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent                  = 0.0) AND
       (SELF\named_unit.dimensions.mass_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.time_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.electric_current_exponent        = 0.0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent     = 0.0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent      = 1.0);
END_ENTITY;
(*
```

Formal propositions:

WR1: the dimensional exponent of luminous intensity shall be equal to one and all the other dimensional exponents shall be equal to zero.

4.14.4.12 plane_angle_unit

A **plane_angle_unit** is the unit in which angles in planes are measured.

EXPRESS specification:

```
*)
ENTITY plane_angle_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent                  = 0.0) AND
       (SELF\named_unit.dimensions.mass_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.time_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.electric_current_exponent        = 0.0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent     = 0.0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent      = 0.0);
END_ENTITY;
```

```
(*
```

Formal propositions:

WR1: all the dimensional exponents shall be equal to zero.

4.14.4.13 solid_angle_unit

A **solid_angle_unit** is the unit in which solid angles are measured.

EXPRESS specification:

```
*)
ENTITY solid_angle_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent                    = 0.0) AND
       (SELF\named_unit.dimensions.mass_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.time_exponent                      = 0.0) AND
       (SELF\named_unit.dimensions.electric_current_exponent          = 0.0) AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent       = 0.0) AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent        = 0.0);
END_ENTITY;
(*
```

Formal propositions:

WR1: all the dimensional exponents shall be equal to zero.

4.14.4.14 area_unit

An **area_unit** is the unit in which the extent of a surface is measured.

EXPRESS specification:

```
*)
ENTITY area_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 2.0)                    AND
       (SELF\named_unit.dimensions.mass_exponent = 0.0)                      AND
       (SELF\named_unit.dimensions.time_exponent = 0.0)                      AND
       (SELF\named_unit.dimensions.electric_current_exponent = 0.0)          AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0)       AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY;
(*
```

Formal propositions:

WR1: the dimensional exponent of length shall be equal to two and all the other dimensional exponents shall be equal to zero.

4.14.4.15 volume_unit

A **volume_unit** is the unit in which the solid content of a body is measured.

EXPRESS specification:

```
*)
ENTITY volume_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 3.0)                    AND
       (SELF\named_unit.dimensions.mass_exponent = 0.0)                      AND
       (SELF\named_unit.dimensions.time_exponent = 0.0)                      AND
       (SELF\named_unit.dimensions.electric_current_exponent = 0.0)          AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0)       AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY;
(*
```

Formal propositions:

WR1: the dimensional exponent of length shall be equal to three and all the other dimensional exponents shall be equal to zero.

4.14.4.16 ratio_unit

A **ratio_unit** is the unit in which the relation between two physical quantities that are of the same kind is measured.

EXPRESS specification:

```
*)
ENTITY ratio_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF\named_unit.dimensions.length_exponent = 0.0)                    AND
       (SELF\named_unit.dimensions.mass_exponent = 0.0)                      AND
       (SELF\named_unit.dimensions.time_exponent = 0.0)                      AND
       (SELF\named_unit.dimensions.electric_current_exponent = 0.0)          AND
       (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0)       AND
       (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
```

```
END_ENTITY;
(*
```

Formal propositions:

WR1: all the dimensional exponents shall be equal to zero.

4.14.4.17 dimensional_exponents

The dimensionality of any quantity can be expressed as a product of powers of the dimensions of base quantities. The **dimensional_exponents** entity defines the powers of the dimensions of the base quantities. All the physical quantities are founded on seven base quantities (ISO 31 (clause 2)).

NOTE – Length, mass, time, electric current, thermodynamic temperature, amount of substance, and luminous intensity are the seven base quantities.

EXAMPLE 84 – A length of 2 millimetres has a length exponent of 1. The remaining exponents are equal to 0.

EXAMPLE 85 – A velocity of 2 millimetres per second has a length exponent of 1 and a time exponent of -1. The remaining exponents are equal to 0.

EXPRESS specification:

```
*)
ENTITY dimensional_exponents;
  length_exponent                    : REAL;
  mass_exponent                      : REAL;
  time_exponent                      : REAL;
  electric_current_exponent          : REAL;
  thermodynamic_temperature_exponent : REAL;
  amount_of_substance_exponent       : REAL;
  luminous_intensity_exponent        : REAL;
END_ENTITY;
(*
```

Attribute definitions:

length_exponent: the power of the length base quantity.

mass_exponent: the power of the mass base quantity.

time_exponent: the power of the time base quantity.

electric_current_exponent: the power of the electric current base quantity.

thermodynamic_temperature_exponent: the power of the thermodynamic temperature base quantity.

amount_of_substance_exponent: the power of the amount of substance base quantity.

luminous_intensity_exponent: the power of the luminous intensity base quantity.

4.14.4.18 derived_unit_element

A **derived_unit_element** is one of the unit quantities which makes up a **derived_unit**.

EXAMPLE 86 – Newtons per square millimetre is a derived unit. It has two elements, Newton whose exponent has a value of 1 and millimetre whose exponent is -2.

EXPRESS specification:

```
*)
ENTITY derived_unit_element;
  unit     : named_unit;
  exponent : REAL;
END_ENTITY;
(*
```

Attribute definitions:

unit: the fixed quantity which is used as the mathematical factor.

exponent: the power that is applied to the **unit** attribute.

4.14.4.19 derived_unit

A **derived_unit** is an expression of units.

EXAMPLE 87 – Newtons per square millimetre is a **derived_unit**.

EXPRESS specification:

```
*)
ENTITY derived_unit;
  elements : SET [1:?] OF derived_unit_element;
WHERE
  WR1 : ( SIZEOF ( elements ) > 1 ) OR
        (( SIZEOF ( elements ) = 1 ) AND ( elements[1].exponent <> 1.0 ));
END_ENTITY;
(*
```

Attribute definitions:

elements: the group of units and their exponents that define the **derived_unit**.

Formal propositions:

WR1: there shall be either more than one member in the **elements** set or the value of the exponent of the single element of the **elements** set shall not be equal to one.

4.14.4.20 global_unit_assigned_context

A **global_unit_assigned_context** is a **representation_context** in which the units apply to all **measure_values** of the correct kind.

NOTE – The kind of a **measure_value** can be deduced from its type name.

EXPRESS specification:

```
*)
ENTITY global_unit_assigned_context
  SUBTYPE OF (representation_context);
  units : SET [1:?] OF unit;
END_ENTITY;
(*
```

Attribute definitions:

units: the units which apply in the **representation_context**.

Informal propositions:

unique_units: each unit shall be a different kind of unit.

4.14.4.21 measure_with_unit

A **measure_with_unit** is the specification of a physical quantity as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY measure_with_unit
  SUPERTYPE OF (ONEOF ( length_measure_with_unit,
                        mass_measure_with_unit,
                        time_measure_with_unit,
                        electric_current_measure_with_unit,
                        thermodynamic_temperature_measure_with_unit,
                        amount_of_substance_measure_with_unit,
                        luminous_intensity_measure_with_unit,
                        plane_angle_measure_with_unit,
                        solid_angle_measure_with_unit,
                        area_measure_with_unit,
                        volume_measure_with_unit,
                        ratio_measure_with_unit ));
  value_component : measure_value;
  unit_component  : unit;
WHERE
  WR1: valid_units (SELF);
END_ENTITY;
(*
```

Attribute definitions:

value_component: the value of the physical quantity when expressed in the specified units.

unit_component: the **unit** in which the physical quantity is expressed.

Formal propositions:

WR1: the unit shall be a valid unit for the kind of measure.

4.14.4.22 length_measure_with_unit

A **length_measure_with_unit** is a **measure_with_unit** where the physical quantity is a length as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY length_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.LENGTH_UNIT' IN TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **length_unit**.

4.14.4.23 mass_measure_with_unit

A **mass_measure_with_unit** is a **measure_with_unit** where the physical quantity is a mass as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY mass_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.MASS_UNIT' IN TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **mass_unit**.

4.14.4.24 time_measure_with_unit

A **time_measure_with_unit** is a **measure_with_unit** where the physical quantity is a time as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY time_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.TIME_UNIT' IN TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **time_unit**.

4.14.4.25 electric_current_measure_with_unit

A **electric_current_measure_with_unit** is a **measure_with_unit** where the physical quantity is a electric_current as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY electric_current_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.ELECTRIC_CURRENT_UNIT' IN TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **electric_current_unit**.

4.14.4.26 thermodynamic_temperature_measure_with_unit

A **thermodynamic_temperature_measure_with_unit** is a **measure_with_unit** where the physical quantity is a thermodynamic_temperature as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY thermodynamic_temperature_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
```

```
  WR1: 'MEASURE_SCHEMA.THERMODYNAMIC_TEMPERATURE_UNIT' IN
       TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **thermodynamic_temperature_unit**.

4.14.4.27 amount_of_substance_measure_with_unit

A **amount_of_substance_measure_with_unit** is a **measure_with_unit** where the physical quantity is a amount_of_substance as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY amount_of_substance_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.AMOUNT_OF_SUBSTANCE_UNIT' IN
       TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **amount_of_substance_unit**.

4.14.4.28 luminous_intensity_measure_with_unit

A **luminous_intensity_measure_with_unit** is a **measure_with_unit** where the physical quantity is a luminous_intensity as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY luminous_intensity_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.LUMINOUS_INTENSITY_UNIT' IN
       TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **luminous_intensity_unit**.

4.14.4.29 plane_angle_measure_with_unit

A **plane_angle_measure_with_unit** is a **measure_with_unit** where the physical quantity is a plane_angle as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY plane_angle_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.PLANE_ANGLE_UNIT' IN
       TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **plane_angle_unit**.

4.14.4.30 solid_angle_measure_with_unit

A **solid_angle_measure_with_unit** is a **measure_with_unit** where the physical quantity is a solid_angle as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY solid_angle_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.SOLID_ANGLE_UNIT' IN TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **solid_angle_unit**.

4.14.4.31 area_measure_with_unit

A **area_measure_with_unit** is a **measure_with_unit** where the physical quantity is a area as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY area_measure_with_unit
  SUBTYPE OF (measure_with_unit);
```

```
WHERE
  WR1: 'MEASURE_SCHEMA.AREA_UNIT' IN TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **area_unit**.

4.14.4.32 volume_measure_with_unit

A **volume_measure_with_unit** is a **measure_with_unit** where the physical quantity is a volume as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY volume_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.VOLUME_UNIT' IN TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **volume_unit**.

4.14.4.33 ratio_measure_with_unit

A **ratio_measure_with_unit** is a **measure_with_unit** where the physical quantity is a ratio as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY ratio_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: 'MEASURE_SCHEMA.RATIO_UNIT' IN TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Formal propositions:

WR1: the unit shall be a **ratio_unit**.

4.14.5 measure_schema function definitions

4.14.5.1 dimensions_for_si_unit

The **dimensions_for_si_unit** function returns the **dimensional_exponents** of the given **si_unit**.

EXPRESS specification:

```
*)
FUNCTION dimensions_for_si_unit (n : si_unit_name) : dimensional_exponents;

  CASE n  OF
    metre          : RETURN (dimensional_exponents
                             (1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    gram           : RETURN (dimensional_exponents
                             (0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    second         : RETURN (dimensional_exponents
                             (0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0));
    ampere         : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0));
    kelvin         : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
    mole           : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0));
    candela        : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    radian         : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    steradian      : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    hertz          : RETURN (dimensional_exponents
                             (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
    newton         : RETURN (dimensional_exponents
                             (1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    pascal         : RETURN (dimensional_exponents
                             (-1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    joule          : RETURN (dimensional_exponents
                             (2.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    watt           : RETURN (dimensional_exponents
                             (2.0, 1.0, -3.0, 0.0, 0.0, 0.0, 0.0));
    coulomb        : RETURN (dimensional_exponents
                             (0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0));
    volt           : RETURN (dimensional_exponents
                             (2.0, 1.0, -3.0, -1.0, 0.0, 0.0, 0.0));
    farad          : RETURN (dimensional_exponents
                             (-2.0, -1.0, 4.0, 1.0, 0.0, 0.0, 0.0));
    ohm            : RETURN (dimensional_exponents
                             (2.0, 1.0, -3.0, -2.0, 0.0, 0.0, 0.0));
    siemens        : RETURN (dimensional_exponents
```

```
                              (-2.0, -1.0, 3.0, 2.0, 0.0, 0.0, 0.0));
    weber           : RETURN (dimensional_exponents
                              (2.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
    tesla           : RETURN (dimensional_exponents
                              (0.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
    henry           : RETURN (dimensional_exponents
                              (2.0, 1.0, -2.0, -2.0, 0.0, 0.0, 0.0));
    degree_Celsius : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
    lumen           : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    lux             : RETURN (dimensional_exponents
                              (-2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    becquerel       : RETURN (dimensional_exponents
                              (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
    gray            : RETURN (dimensional_exponents
                              (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    sievert         : RETURN (dimensional_exponents
                              (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
  END_CASE;
END_FUNCTION;
(*
```

Argument definitions:

n : (input) the name of the **unit** for which the **dimensional_exponents** will be returned.

4.14.5.2 derive_dimensional_exponents

This function determines the dimensional exponents of a unit. For named units the **dimensions** attribute is returned and for derived units the dimensional exponents are calculated from its elements.

EXPRESS specification:

```
*)
FUNCTION derive_dimensional_exponents (x : unit) : dimensional_exponents;

  LOCAL
    i      : INTEGER;
    result : dimensional_exponents :=
             dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
  END_LOCAL;

  IF 'MEASURE_SCHEMA.DERIVED_UNIT' IN TYPEOF(x) THEN -- x is a derived unit
    REPEAT i := LOINDEX(x.elements) TO HIINDEX(x.elements);

      result.length_exponent                    :=
```

```
        result.length_exponent +
        (x.elements[i].exponent *
         x.elements[i].unit.dimensions.length_exponent);

      result.mass_exponent                    :=
        result.mass_exponent +
        (x.elements[i].exponent *
         x.elements[i].unit.dimensions.mass_exponent);

     result.time_exponent :=
        result.time_exponent +
        (x.elements[i].exponent *
         x.elements[i].unit.dimensions.time_exponent);

      result.electric_current_exponent        :=
        result.electric_current_exponent +
        (x.elements[i].exponent *
         x.elements[i].unit.dimensions.electric_current_exponent);

      result.thermodynamic_temperature_exponent :=
        result.thermodynamic_temperature_exponent +
        (x.elements[i].exponent *
         x.elements[i].unit.dimensions.thermodynamic_temperature_exponent);

      result.amount_of_substance_exponent     :=
        result.amount_of_substance_exponent +
        (x.elements[i].exponent *
         x.elements[i].unit.dimensions.amount_of_substance_exponent);

      result.luminous_intensity_exponent      :=
        result.luminous_intensity_exponent +
        (x.elements[i].exponent *
         x.elements[i].unit.dimensions.luminous_intensity_exponent);

    END_REPEAT;
  ELSE -- x is a unitless or a named unit
    result := x.dimensions;
  END_IF;
  RETURN (result);
END_FUNCTION;
(*
```


Argument definitions:

x: (input) the unit that the **dimensional_exponents** are being derived from.

4.14.5.3 valid_units

The **valid_units** function validates a **measure_with_unit**. If the unit of the **measure_with_unit** is valid then the function returns TRUE, otherwise it returns FALSE.

EXPRESS specification:

```
*)
FUNCTION valid_units ( m : measure_with_unit ) : BOOLEAN ;

  IF 'MEASURE_SCHEMA.LENGTH_MEASURE' IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
       dimensional_exponents ( 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;

  IF 'MEASURE_SCHEMA.MASS_MEASURE' IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
       dimensional_exponents ( 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;

  IF 'MEASURE_SCHEMA.TIME_MEASURE' IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
       dimensional_exponents ( 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;

 IF 'MEASURE_SCHEMA.ELECTRIC_CURRENT_MEASURE' IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
       dimensional_exponents ( 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;

  IF 'MEASURE_SCHEMA.THERMODYNAMIC_TEMPERATURE_MEASURE'
  IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
       dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;

  IF 'MEASURE_SCHEMA.AMOUNT_OF_SUBSTANCE_MEASURE' IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
       dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;

  IF 'MEASURE_SCHEMA.LUMINOUS_INTENSITY_MEASURE' IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
       dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0 ) THEN
```

```
    RETURN (FALSE);
  END_IF;
END_IF;

IF 'MEASURE_SCHEMA.PLANE_ANGLE_MEASURE' IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
     dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;

IF 'MEASURE_SCHEMA.SOLID_ANGLE_MEASURE' IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
     dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;

IF 'MEASURE_SCHEMA.AREA_MEASURE' IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
     dimensional_exponents ( 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;

IF 'MEASURE_SCHEMA.VOLUME_MEASURE' IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
     dimensional_exponents ( 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;

IF 'MEASURE_SCHEMA.RATIO_MEASURE' IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
     dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;

IF 'MEASURE_SCHEMA.POSITIVE_LENGTH_MEASURE' IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
     dimensional_exponents ( 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;

IF 'MEASURE_SCHEMA.POSITIVE_PLANE_ANGLE_MEASURE' IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
     dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
```

```
  RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

m: (input) the candidate **measure_with_unit** that is to be checked.

EXPRESS specification:

```
*)
END_SCHEMA; -- measure_schema
(*
```