TECHNICAL SPECIFICATION

ISO/IEC TS 19763-13

First edition
2016-12-01

Information technology — Metamodel framework for interoperability (MFI) —

Part 13: Metamodel for form design registration

Technologies de l'information — Cadre du métamodèle pour l'interopérabilité (MFI) —

Partie 13: Métamodèle pour l'enregistrement de la conception des formulaires

Reference number
ISO/IEC TS 19763-13:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 32, *Data management and interchange*.

A list of all parts in the ISO/IEC 19763 series can be found on the ISO website.

Introduction

There is an increasing demand for systems to interoperate by exchanging data, and for data to be reused outside of the original context of its collection. For data exchange or reuses to be meaningful, the business information requirements that are met by the data stored in these systems must be understood so that suitable data exchange mechanisms can be developed and interpretation of the data is reliable.

Not only does this require a clear understanding of the meaning of the data, it also frequently requires the coordination of data capture. Where data input is manual, the definitive source of data semantics is the design of the data entry form. Indeed if we do not understand the encoding of knowledge in the database schema or we suspect some anomaly in the data captured, we inspect the original form and the context of its use. Furthermore, if we wish to gather interoperable data, it is frequently necessary to harmonize aspects of form design before information systems are developed and data is captured. However, there is no abstract, universal metamodel for form designs that supports the registration and comparison or harmonization of form designs and faithful implementation of these designs in information systems. This is the intent of this document.

The Oxford English dictionary defines a form as "a formulary document with blanks for the insertion of particulars". Other ISO definitions of a form include ISO 5127, "document (printed or otherwise produced), with pre-designated spaces for the recording of specific information", and ISO 9241-143, "structured display of fields and other user-interface elements that the user reads, fills in, selects entries for (e.g. through check boxes or radio buttons) or modifies". While we recognize these definitions, none precisely matches the needs of this document. Thus, we will define a form as a structured collection of spaces, suitable instructions and rules that support the collection of specific information that may be subsequently compared and processed in a routine fashion. A form design is thus a description of a particular form such that it may be rendered in any suitable information system, and the metamodel for registration of form designs contained within this document describes the attributes that are necessary to represent the semantics and syntax of form designs.

Given a standard metamodel for the registration of form designs, ISO/IEC 19763 Metamodel framework for interoperability (MFI) and ISO/IEC 11179 Metamodel for metadata registries provide important facilities for the creation and annotation of form designs. ISO/IEC 19763 supports the registration of form designs and section elements as models and model elements, provides facilities to record associations between the components of two or more form design, particularly derivation, specialization, extension and reuse, and allows the association of form designs with the data models that are used to store data captured by their instances. ISO/IEC 11179 provides classes and types that support the identification, naming, registration and administration of form designs and supporting documents, and provides a model either for an associated, standardized question bank or a rich source of question-level metadata attributes with which to explain the meaning of individual data items. When used together, the International Standards can support the rapid design and reuse of form designs, wrap and hide the complexity of semantic annotation from subject matter experts, and provide a ready reference of associations and transformations for users seeking to collect and use interoperable data.

This document does not supplant or replace computer languages such as XForms, Windows Forms, Adobe Forms or relevant parts of HTML, which describe how a form design is implemented, and is deliberately devoid of domain or content specific semantics to ensure wide applicability. However, given the universal applicability of forms, it should be of no surprise that elements of the model can be recognized in many forms standards. Some of these have been mapped to this document in Annex A to Annex E.

Forms may be printed on paper, or encoded in electronic format. Electronic forms may be rendered natively in standard formats such as HTML, XForms or PDF, or propriety ones such as Windows forms, Cocoa or Java Swing. They may also be implemented in a common survey framework such as Survey Monkey or Lime Survey. Despite this diversity, it is eminently possible to create forms in different formats that support the same comparisons and downstream processing *provided the spaces and instructions share the same semantic intent*. Such a collection of forms could be said to share the same *design*. A model that is adequate to record these *form designs* is the subject of this document.

Information technology — Metamodel framework for interoperability (MFI) —

Part 13: Metamodel for form design registration

1 Scope

The primary purpose of the ISO/IEC 19763 series is to specify a metamodel framework for interoperability. This document specifies a metamodel for registering form designs.

This document provides a metamodel to describe the structure and semantics of an implemented form devoid of any specific, domain semantics, e.g. in healthcare, social science, e-government and e-business, or representation format so that data may be faithfully exchanged between systems and system components, and associations expressed between sets of form designs whose data may be compared, joined or composed for analysis.

2 Normative references

There are no normative references in this document.

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19763-1, ISO/IEC 19763-10, ISO/IEC 11179-3 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1.1
attachment
digital object that is required as a *response* (3.1.15) to a *question* (3.1.14) on a *form* (3.1.9)

Note 1 to entry: Used to indicate that the response to a question includes a file on an accessible file-system that will be loaded when the form transaction is complete.

3.1.2
combinator
operator that joins two *constraints* (3.1.6) (to make a binary constraint) returning a result based upon both

EXAMPLE Conjunction, disjunction, implication.

3.1.3
compliance rule
<form template> specification for some aspect of a *form design* (3.1.10) that shall be satisfied for that design to be a correct implementation of a *form template* (3.1.11)

3.1.4
completed form
form (3.1.9) for which *responses* (3.1.15) have been completed as required according to its *instructions* (3.1.12) and *rules* (3.1.16)

3.1.5
consequence
expression (3.1.7) that sets or specifies some property of an element of a *form design* (3.1.10) when its related *constraint* (3.1.6) evaluates to true

3.1.6
constraint
<form registration> *expression* (3.1.7) about *form design* (3.1.10) elements that evaluates to a Boolean value

3.1.7
expression
statement that evaluates to a string or numeric value

3.1.8
field
space on a *form* (3.1.9) for the recording of a *response* (3.1.15)

3.1.9
form
document or human interface comprising a structured collection of *fields* (3.1.8), suitable *instructions* (3.1.12) and *rules* (3.1.16) that support the collection of specific information that may be subsequently compared and processed in a routine fashion

3.1.10
form design
specification for the creation of equivalent *forms* (3.1.9) in different languages, applications and media

3.1.11
form template
partial *form design* (3.1.10) that establishes a pattern for the creation of other form designs

Note 1 to entry: A form template will often have empty or incomplete form sections with instructions describing what kind of questions are required to create a completed design.

3.1.12
instruction
sentence that directs a person in some aspect of the completion or submission of a *form* (3.1.9)

3.1.13
owl:sameAs
property of the Web Ontology Language that indicates that individuals in an OWL DL ontology refer to the same thing, or in OWL Full to additionally indicate that two classes are equal

Note 1 to entry: See http://www.w3.org/TR/owl-ref/#sameAs-def.

3.1.14
question
sentence worded or expressed so as to elicit information from a person

3.1.15
response
information elicited from a person by a *question* (3.1.14)

3.1.16
rule
principle guiding the behaviour of some aspect of a *form* (3.1.9)

3.1.17
section
subcomponent of a *form* (3.1.9) whose contained *questions* (3.1.14), *instructions* (3.1.12) and *rules* (3.1.16) share a common purpose, meaning or context

3.1.18
skos:related
semantic relation asserting that the object of the labelled relationship is related to the subject

Note 1 to entry: See http://www.w3.org/TR/skos-reference.

3.2 Abbreviated terms

MFI Core and mapping

ISO/IEC 19763-10, Information technology — Metamodel framework for interoperability (MFI) — Part 10: Core model and basic mapping

MDR Metamodel

ISO/IEC 11179-3, Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes

MFI Form design registration

Information technology — Metamodel framework for interoperability (MFI) — Part 13: Metamodel for form design registration

4 Conformance

4.1 General

An implementation claiming conformance with this document shall support the metamodel specified in Clause 5, depending on a degree of conformance as described below.

4.2 Degrees of conformance

4.2.1 General

The distinction between "strictly conforming" and "conforming" implementations is necessary to address the simultaneous needs for interoperability and extensions. This document describes specifications that promote interoperability. Extensions are motivated by needs of users, vendors, institutions and industries, but are not specified by this document.

A strictly conforming implementation may be limited in usefulness but is maximally interoperable with respect to this document. A conforming implementation may be more useful, but may be less interoperable with respect to this document.

4.2.2 Strictly conforming implementation

A strictly conforming implementation

a) shall support the metamodel specified in Clause 5, and

b) shall not support any extensions to the metamodel specified in Clause 5.

4.2.3 Conforming implementation

A conforming implementation

a) shall support the metamodel specified in Clause 5, and

b) may support extensions to the metamodel specified in Clause 5 that are consistent with the metamodel and the MDR mapping package in Clause 5.

4.2.4 Implementation Conformance Statement (ICS)

An implementation claiming conformance with this document shall include an Implementation Conformance Statement stating

a) whether it is a strictly conforming implementation or a conforming implementation (see 4.2.2), and

b) what extensions are supported if it is a conforming implementation (see 4.2.3).

Conformance statements for systems that implement this document shall additionally describe the languages used to convey Rules, and the relationship types available for the Mapping_Relation class.

5 Structure of MFI form design registration

5.1 Overview of MFI form design registration

Figure 1 shows the metamodel for the registration of form designs.

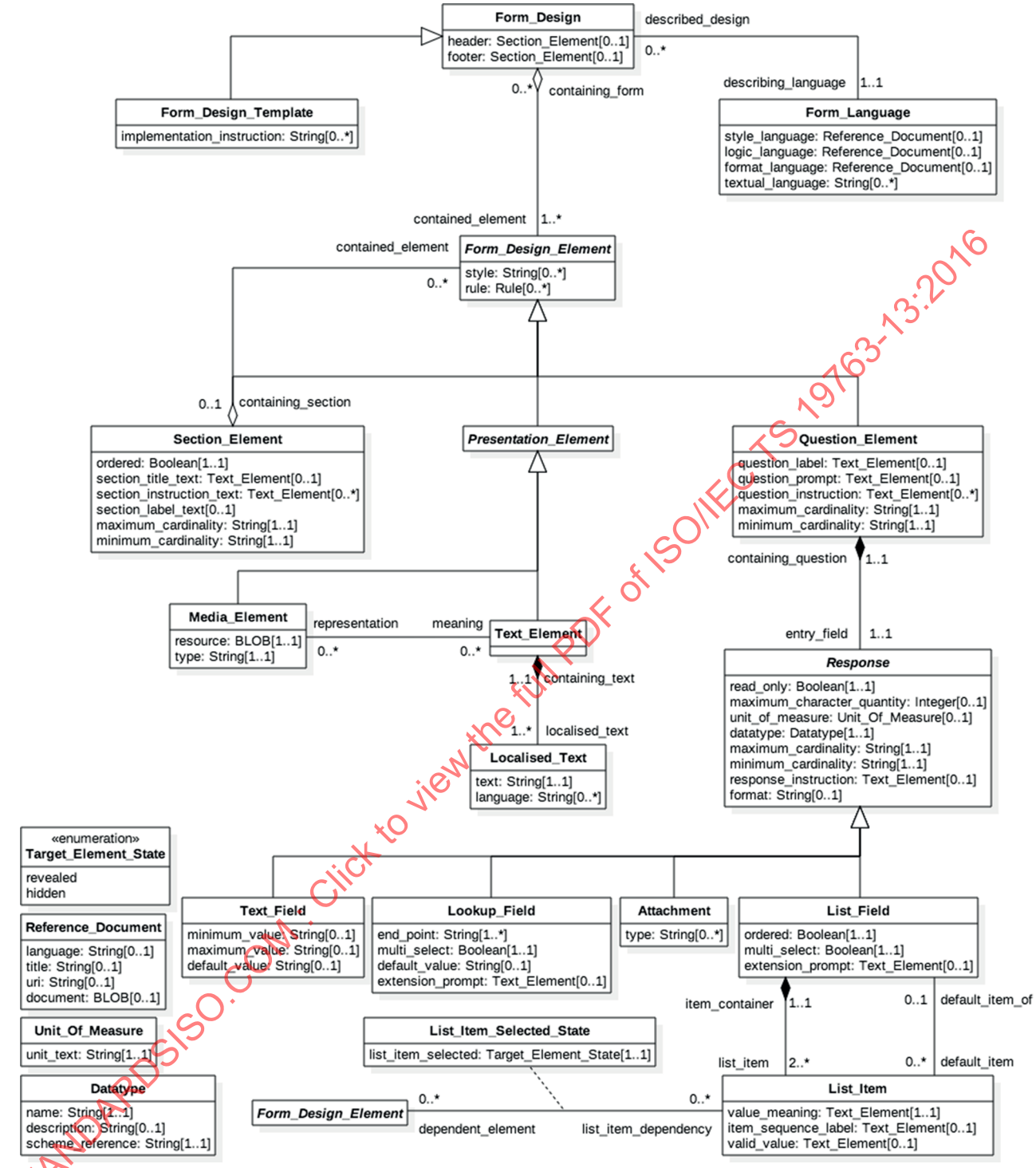

Figure 1 — Form design metamodel

Forms have questions and sections that are constrained or unavailable for completion dependent upon the answers given to earlier questions. Figure 2 is a model for the rule language used to describe such dependencies between form elements: textual expressions in this language are used to complete the *rule* attribute of the **Form_Design_Element** class.

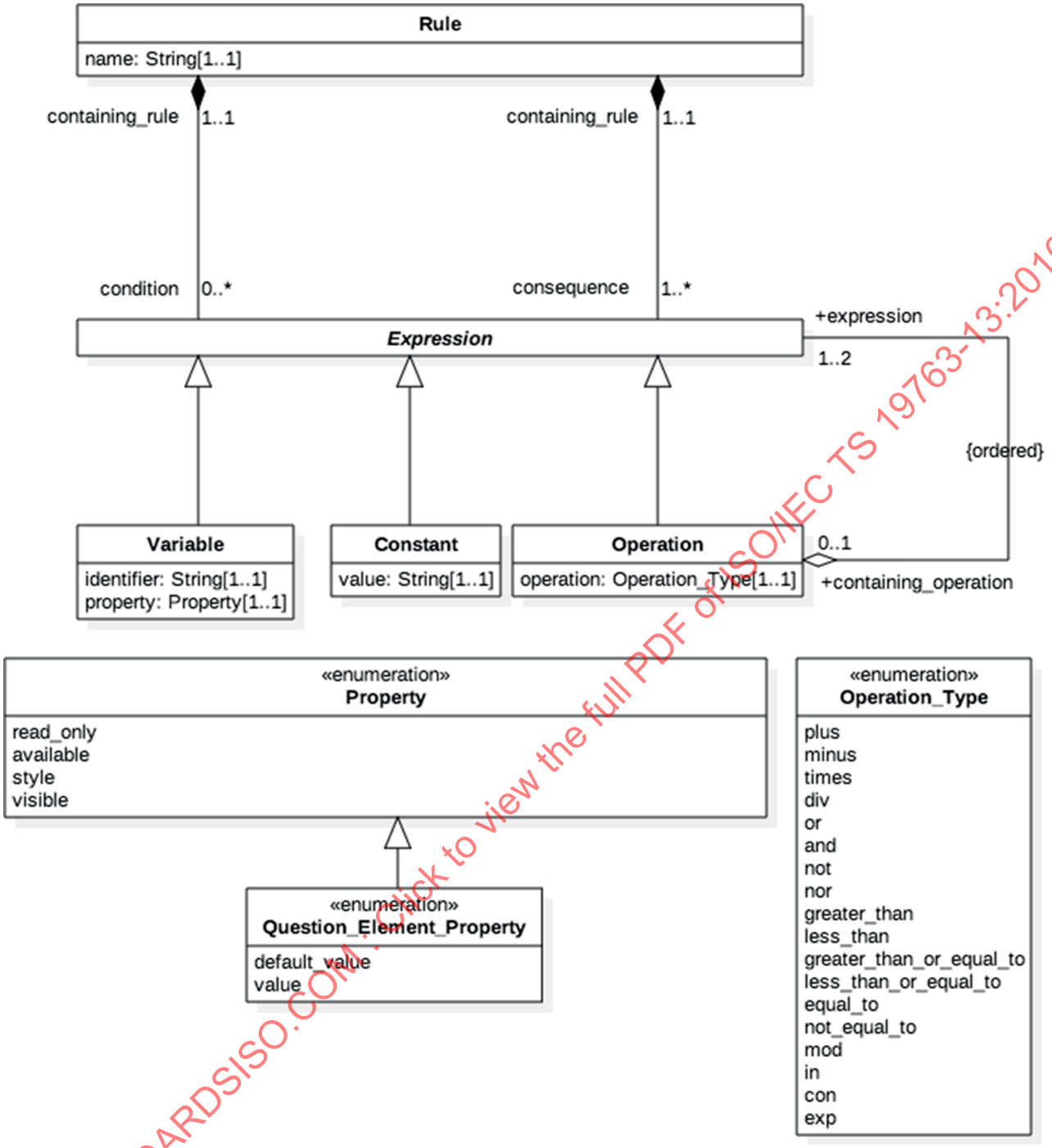

Figure 2 — Rule

The metamodel for information model registration comprises the following metaclasses:

Attachment_Field

Constant

Datatype

Expression

Form_Design

Form_Design_Element

Form_Design_Template

Form_Design_Language

List_Field

List_Item

List_Item_Selected_State

Localised_Text

Lookup_Field

Media_Element

Operation

Presentation_Element

Question_Element

Response

Reference_Document

Rule

Section_Element

Text_Element

Text_Field

Variable

Unit_of_Measure

The purpose and use of the metamodel is described in detail in Annex A. Detailed specifications of the metaclasses are provided in Annex B.

5.2 Relationship of metaclasses to the MDR Metamodel

As explained in ISO/IEC 19763-10, instances of the metaclasses defined in this subclause may be extended by the types defined in the MDR Metamodel as follows.

— **Form_Design** may be extended as an **Identified_Item**, **Designatable_Item**, **Registered_Item**, **Administered_Item** and **Classifiable_Item**.

— **Form_Design_Element** may be extended as an **Identified_Item**, **Designatable_Item** and **Classifiable_Item**.

— Any instance of a **Form_Design_Element** may be mapped to an instance of a **Concept**.

— Any instance of a **Question_Element** may be mapped to an instance of a **Data_Element**.

— **List_Item** may be extended as an **Identified_Item**; any instance of which may be mapped to a **Concept** and/or **Permissible_Value**.

— **Rule** may be extended as an **Identified_Item** and **Designatable_Item**.

5.3 Details provided in each metaclass definition

For each metaclass, the following details are shown:

— a definition that describes the role or significance of instances of the metaclass;

— the name of its immediate supertype;

— any alternative names (synonyms or aliases) for the metaclass;

— a list of attributes;

— a list of references.

For each attribute, the following details are shown:

a) the name of the attribute; where the attribute is one that is provided by the type defined in the MDR metamodel by which when instances of the metaclass are extended, the name is italicized;

b) the datatype for values of the attribute;

c) the multiplicity of the attribute;

d) a description that describes the role or significance of values of the attribute.

For each reference, the following details are shown:

— the name of the reference; this is the role name that describes the role played by the referenced metaclass with respect to the association identified by this reference;

— the name of the referenced metaclass;

— the multiplicity of the reference;

— a description that describes the role or significance of the instance, or instances, of the referenced metaclass with respect to an instance of this metaclass;

— the name of the reference in the referenced metaclass that provides the inverse definition for the association;

— an indication as to whether this metaclass is responsible for the maintenance of the association, i.e. the precedence of the metaclass with respect to the association.

5.4 Basic Types and Enumerations in MFI form design registration

5.4.1 General

Basic Types specify common datatypes for use in the metaclasses. A datatype is a set of distinct values, characterized by properties of those values and by operations on those values (see ISO/IEC 11404). The datatypes used in the specification of the **metaclasses** (see 5.5) are restricted to Boolean, Integer, Date, Value, Sign, Postal_Address, String, Natural_Range, Datetime, String, Notation and Phone_Number (MDR Metamodel 6.2.1 Overview of Basic Types). The types used in the metaclasses are based on this core set of types, with a single addition of the type Binary Large Object (BLOB), and any compliant implementation of a metadata registry should include an implementation of the semantics specified in these core types.

NOTE These datatypes are used in specification of the metaclass attributes themselves, and are not intended to constrain the datatypes that may be used in specifying Response datatypes.

Enumerations specify the list of value for use with metaclass attributes.

For each enumeration, the following details are shown:

— the name of the referenced enumeration;

— a description of the enumeration;

— the datatype of the values in the enumeration;

— the name of each value in the enumeration;

— a description of the semantics of each enumeration value;

— the name of the metaclass where this enumeration is used;

— the name of the attribute where this enumeration is used.

5.4.2 Property

Property is an enumeration of values listing properties of a **Presentation_Element**, **Section_Element**, **Question_Element** or a **List_Item** that may be addressed by a **Rule** (see Figure 2).

Datatype

String

Value	Description
read_only	Indicates that the **Form_Design_Element** read_only property is to be tested or set as part of an **Expression** in a **Rule**
available	Indicates that the **Form_Design_Element** available property is to be tested or set as part of an **Expression** in a **Rule** The state of the available property may also be set by a **List_Item** that has a dependent_element association with the respective **Form_Design_Element**.
style	Indicates that a **Form_Design_Element** style property is to be tested or set as part of an **Expression** in a **Rule**
visible	Indicates that a **Form_Design_Element's** visibility to the user interacting with the form is to be tested or set as part of an **Expression** in a **Rule**

5.4.3 Question_Element_Property

Question_Element_Property is an enumeration of values listing additional properties of a **Question_Element** that may be addressed in a **Rule** (see Figure 2).

Supertype

Property

Datatype

String

Value	Description
default_value	Indicates that the **Question_Element** default_value property is to be tested or set as part of an **Expression** in a **Rule**
value	Indicates that the **Question_Element** value property is to be tested or set as part of an **Expression** in a **Rule**

5.4.4 Target_Element_State

Target_Element_State is an enumeration of values listing the possible states that a dependent **Form_Design_Element** may take when a **List_Item** is selected.

Datatype

String

Value	Description
revealed	Indicates that the dependent **Form_Design_Element** should be visible or available for input when the **List_Item** is selected
hidden	Indicates that the dependent **Form_Design_Element** should be hidden or unavailable for input when the **List_Item** is selected

5.4.5 Operation_Type

Operation_Type is an enumeration of values describing the operation between two items in an **Expression** (see Figure 2).

Datatype

String

Value	Description
plus	Indicates the mathematical addition operation between the two items in the **Expression**
minus	Indicates the mathematical subtraction operation between the two items in the **Expression**
times	Indicates the mathematical multiplication operation between the two items in the **Expression**
div	Indicates the mathematical division operation between the two items in the **Expression**
or	Indicates a logical “or” between the two items in the **Expression**
and	Indicates a logical “and” between the two items in the **Expression**
not	Indicates a logical “not” between the two items in the **Expression**
nor	Indicates a logical “nor” between the two items in the **Expression**
greater_than	Indicates the mathematical “greater-than” operation between the two items in the **Expression**
less_than	Indicates the mathematical “less-than” operation between the two items in the **Expression**
greater_than_or_equal_to	Indicates the mathematical “greater-than or equal-to” operation between the two items in the **Expression**
less_than_or_equal_to	Indicates the mathematical “less-than or equal-to” operation between the two items in the **Expression**
equal_to	Indicates the mathematical “equals” operation between the two items in the **Expression**
not_equal_to	Indicates the mathematical “not equal to” operation between the two items in the **Expression**
mod	Indicates the mathematical modulo operation between the two items in the **Expression**

in	Indicates an operation between two items in the **Expression** where one item is a list. It evaluates to "true" if the item is one of the enumerations. For example, an expression "condition person.ecog IN (0,1,2) consequence person.eligible-for-trial = true" would set the person.eligible-for-trial question to "true" if the value entered for the person.ecog question is 0, 1 or 2.
con	Indicates the programmatic "string concatenation" operation between the two items in the **Expression**
exp	Indicates the programmatic "exponent" operation between the two items in the **Expression**

5.5 Metaclasses in MFI for form design registration

5.5.1 Form_Design

Form_Design is a metaclass, each instance of which represents the design of a specific form, which is formulary document with blanks for the insertion of particulars (see Figure 1).

5.5.2 Form_Design_Language

Form_Design_Language is a metaclass, each instance of which represents the selection of languages used to express aspects of the design of the associated **Form_Design** (see Figure 1).

5.5.3 Form_Design_Template

Form_Design_Template is a metaclass, each instance of which represents a specific form template which is a partially complete form design intended to guide the creation of similar form designs (see Figure 1).

Superclass

Form_Design

Attribute	Datatype	Multiplicity	Description
implementation_instruction	String	0..*	An optional instruction describing how to instantiate a valid **Form_Design** instance from this **Form_Design_Template** instance

5.5.4 Form_Design_Element

Form_Design_Element is an abstract metaclass, each instance of which represents some component of an instance of the class **Form_Design** (see Figure 1).

SuperClass

Model_Element (defined in MFI Core and mapping)

Attribute	DataType	Multiplicity	Description
style	String	0..*	An optional set of statements in some style language about this element and its contained elements that declares layout properties such as emphasis, colour, font size, typeface, line-style and position where the maximum multiplicity is unbounded
rule	**Rule**	0..*	A set of expressions that describe functional dependencies and constraints upon data entry relevant to the semantics of the completed form

Reference	Class	Multiplicity	Description	Inverse	Precedence
contained_section	**Section_Element**	0..*	The optional set of **Section_Element** instances contained by this instance of **Form_Design_Element** where the maximum multiplicity is unbounded	contained_element	yes
containing_form	**Form_Design**	0..*	The instance of **Form_Design** within which this **Form_Design_Element** instance is contained where the maximum multiplicity is unbounded	contained_element	no

NOTE It is not intended that a **Question_Element** can contain a **Section_Element**, although some **Presentation_Element** instances could, e.g. a box. It is preferred to add a stylesheet reference to the section to include a box around contained elements.

5.5.5 Presentation_Element

Presentation_Element is an abstract metaclass, each instance of which is a presentation component of the form, for example some image, box, line or text (see Figure 1).

Presentation element instances should have no notable semantic context unless they are explicitly linked to a **Section_Element**, **Question_Element**, **Response** or **List_Item** instance.

A **Text_Element** instance may be associated with a **Question_Element** instance so as to make it the Prompt for a question on the form, or a **Media_Element** instance may be associated with some semantic **Text_Element** instance as a representation of the concept that **Text_Element** instance conveys.

Superclass

Form_Design_Element

5.5.6 Section_Element

Section_Element is a metaclass, each instance of which is a section of an instance of the class Form_Design that describes a structural association between a set of **Form_Design_Element** instances (see Figure 1).

Superclass

Form_Design_Element

Attribute	DataType	Multiplicity	Description
ordered	Boolean	1..1	A flag that indicates if the order of child **Form_Design_Element** instances is semantically significant where the maximum multiplicity is one. NOTE There are no requirements for a specific implementation of ordering in a conformant system. Implementers are free to use the natural capabilities of their implementation environment to support ordering.
Section_title_text	**Text_Element**	0..1	An optional attribute that declares a particular **Text_Element** as the section title where the maximum multiplicity is one
section_instruction_text	**Text_Element**	0..*	An optional association that declares a particular **Text_Element** as a section_instruction with a maximum multiplicity of unbounded
section_label_text	**Text_Element**	0..1	An optional attribute that specifies a sequence label for this section with a maximum multiplicity of one
maximum_cardinality	String	1..1	A mandatory attribute specifying the maximum number of repetitions of this **Section_Element** instance that are allowed on the completed form that this **Form_Design** instance describes
minimum_cardinality	String	1..1	A mandatory attribute specifying the minimum number of repetitions of this **Section_Element** instance that are allowed on the completed form that this **Form_Design** instance describes

Reference	Class	Multiplicity	Description	Inverse	Precedence
contained_element	**Form_Design_Element**	0..*	The optional set of **Form_Design_Element** instances contained by this **Section_Element** instance where the maximum multiplicity is unbounded	contained_section	yes

5.5.7 Media_Element

Media_Element is a metaclass, each instance of which represents some image, audio or video element presented within a form (see Figure 1).

Superclass

Form_Design_Element

Attribute	DataType	Multiplicity	Description
resource	BLOB	1..1	A mandatory attribute containing the media file that is to be displayed on the form with a maximum multiplicity of one
type	String	1..1	A mandatory attribute conveying the mime-type of the media file that is to be displayed with a maximum multiplicity of one

Reference	Class	Multiplicity	Description	Inverse	Precedence
meaning	**Text_Element**	0..*	An optional association to a particular instance of a **Text_Element** that describes the meaning of the media element with a maximum multiplicity of unbounded	representation	yes

5.5.8 Text_Element

Text_Element is a metaclass, each instance of which is a textual presentation element of a form intended to instruct or explain to the user of the form what the data should mean, how it should be completed and any actions that should be taken with the completed form (see Figure 1).

Superclass

Presentation_Element

Reference	Class	Multiplicity	Description	Inverse	Precedence
localised_text	**Localised_Text**	1..*	A mandatory association to an instance of a **Localised_Text** class which contains the displayed text together with the natural, human language of that text where the maximum multiplicity is unbounded	containing_text	yes
representation	**Media_Element**	0..*	An optional association to a **Media_Element** for representation of a particular **Text_Element** where the maximum multiplicity is unbounded	meaning	no

5.5.9 Localised_Text

Localised_Text is a metaclass, each instance of which represents a pairing of a text string to be displayed to the user of the form and its human language designator. **Localised_Text** provides the capability to register a set of semantically identical forms, and forms that display multiple human languages in a singular item (see Figure 1).

Attribute	DataType	Multiplicity	Description
text	String	1..1	The text string itself
language	String	0..*	A language designation in ISO 639-3 which identifies the language of the associated text entry This attribute may be omitted if the form is mono-lingual and either the language is declared using the textual_language attribute of the **Form_Design_Language** class (see 5.5.2), or if the language of the form is the primary language of the registry (see MDR Metamodel 8.1.2.9.2.7). NOTE Some, mainly paper, forms are multilingual by having alternate languages in the same question and thus may require multiple language identifiers, e.g. “Q1. Surname/Nom de famille/Familien-oder Nachname” …

Reference	Class	Multiplicity	Description	Inverse	Precedence
containing_text	**Text_Element**	1..1	A mandatory association to the containing **Text_Element** instance	localised_text	no

5.5.10 Question_Element

Question_Element is a metaclass, each instance of which represents a question in a **Form_Design** instance (see Figure 1).

Question_Element instances and the values of their attributes may be associated or sourced from a related **MDR Metamodel Data_Element** instance.

Superclass

Form_Design_Element

Attribute	DataType	Multiplicity	Description
question_label	**Text_Element**	0..1	An optional attribute, each instance represents the label or number of this question instance with a maximum multiplicity of one. NOTE 1 An attribute of either question_label or question_prompt is required for a complete form design instance.
question_prompt	**Text_Element**	0..1	An optional attribute, each instance of which represents the actual question text associated with this **Question_Element** instance with a maximum multiplicity of one. NOTE 2 An attribute of either question_label or question_prompt is required for a complete form design instance.
question_ instruction	**Text_Element**	0..*	An optional attribute, each instance of which represents some additional instructional text for this **Question_Element** instance with a maximum multiplicity of unbounded.
maximum_ cardinality	String	1..1	A mandatory attribute specifying the maximum number of times this question may be answered when this form design instance is presented to the user as a form.
minimum_ cardinality	String	1..1	A mandatory attribute specifying the minimum number of times this question may be answered when this form design instance is presented to the user as a form.

Reference	Class	Multiplicity	Description	Inverse	Precedence
entry_field	**Response**	1..1	A mandatory association to a **Response** instance with a maximum multiplicity of one	containing_ question	yes

5.5.11 Response

Response is an abstract metaclass each instance of which represents that part of a **Question_Element** which allows entry of a value (see Figure 1).

Several attributes correspond to those of a **MDR metamodel Data_Element**, and may be set by mapping to the data element in an accessible registry.

NOTE A design admits no notion of the value of the response, though clearly any instance of response will have. See B.5 for discussion of the relationship between the design model, the design instance and the form instance itself.

Attribute	DataType	Multiplicity	Description
read_only	Boolean	1..1	An indicator of whether the value of the **Response** can be edited, where the maximum multiplicity is one
maximum_ character_ quantity	Integer	0..1	An optional maximum number of characters that the **Response** may accept where the maximum multiplicity is one
unit_of_measure	**Unit_of_Measure**	0..1	An optional textual name for the units when the value of the **Response** is measured, where the maximum multiplicity is one
datatype	**Datatype**	1..1	String that identifies the type of data to be stored for the answer where the maximum multiplicity is one
maximum_ cardinality	String	1..1	The mandatory maximum number of responses that may be given to this question where the maximum multiplicity is one
minimum_ cardinality	String	1..1	The optional number of answers to the question that may be provided where the maximum multiplicity is one.
response_ instruction	**Text_Element**	0..1	An optional **Text_Element** instance that provides some instruction associated with the **response** with a maximum multiplicity of one
format	String	0..1	An optional template for the structure of the presentation of the value(s), for example, YYYY-MM-DD for a date. Maximum multiplicity is one. The format_language should be specified in the **Form_Design_Language** class.

Reference	Class	Multiplicity	Description	Inverse	Precedence
containing_ question	**Question_Element**	1..1	The **Question_ Element** to which this **Response** belongs where the maximum multiplicity is one	entry_field	no

5.5.12 Attachment

Attachment is a metaclass, each instance of which represents a field which receives a digital object or an instruction to include a physical attachment in response to a question.

Superclass

Response

Attribute	DataType	Multiplicity	Description
type	String	0..*	An optional attribute describing the type of electronic attachment. For example, The Internet Media Type.

5.5.13 Text_Field

Text_Field is a metaclass, each instance of which represents a field on a form into which any text characters may be entered, subject to the pattern and length constraints.

Superclass

Response

Attribute	DataType	Multiplicity	Description
minimum_value	String	0..1	An optional string representing the lower limit in the range of values that are an acceptable response to the question that is represented by the containing **Question_Element** instance, where the maximum multiplicity is one
maximum_value	String	0..1	An optional string representing the upper limit in the range of values that are an acceptable response to the question that is represented by the containing **Question_Element** instance, where the maximum multiplicity is one
default_value	String	0..1	An optional default value for the response, where the maximum multiplicity is one

5.5.14 Lookup_Field

Lookup_Field is a metaclass, each instance of which represents a field which, like a **List_Field**, has a valid list of answers from a defined domain, but where the members of the domain vary with time and between implementations, e.g. a view providing a valid set of active customer IDs for a sales order system, a terminology approved for tagging an experimental result, an open issue identifier lookup in bug tracking software, the set of identifiers of your friends that you might use to tag a post on a social media site.

Superclass

Response

Attribute	DataType	Multiplicity	Description
end_point	String	1..*	The location of the endpoint providing the value, a service or function call, a URI call that returns the value list where the maximum multiplicity is unbounded
multi_select	Boolean	1..1	A mandatory flag that indicates if more than one option from the set of allowed responses may be selected, with a maximum multiplicity of one
default_value	String	0..1	An optional default value for the response, where the maximum multiplicity is one
extension_prompt	Text_Element	0..1	An optional association with a Prompt to provide the ability to make a single free text entry instead of the items in the **Lookup_Field**. For example, "Other:" or "Specify:"

5.5.15 List_Field

List_Field is a metaclass, each instance of which represents a field for which a set of valid answers are predefined.

Superclass

Response

Attribute	DataType	Multiplicity	Description
ordered	Boolean	1..1	A mandatory flag that indicates whether or not the order of child **List_Item** instances is semantically significant, where the maximum multiplicity is one. NOTE There are no requirements for a specific implementation of ordering in a conformant system. Implementers are free to use the natural capabilities of their implementation environment to support ordering.
multi_select	Boolean	1..1	A mandatory flag that indicates if more than one option from the set of allowed responses may be selected, with a maximum multiplicity of one
extension_prompt	Prompt	0..1	An optional attribute that indicates that the set of allowed responses may be extended by a user-entered value, and which specifies the prompt that is to be shown to the user, e.g. "Other:" or "Specify:"

Reference	Class	Multiplicity	Description	Inverse	Precedence
default_item	**List_Item**	0..*	An optional set of pre-defined list items that are automatically selected as answers to the question unless overridden by the user, where the maximum multiplicity is number of **List_Item** instances	default_ item_of	yes
list_Item	**List_Item**	2..*	The set of pre-defined list items that are allowed answers to the question where the minimum multiplicity is two and the maximum multiplicity is unbounded	item_ container	yes

5.5.16 List_Item

List_Item is a metaclass, each instance of which represents an available answer for a **List_Field** instance.

Superclass

none

Attribute	DataType	Multiplicity	Description
value_meaning	**Text_Element**	1..1	A mandatory attribute that represents the text read by the user of the form when selecting the **List_Item** instance with a maximum multiplicity of one
item_sequence_ label	**Text_Element**	0..1	An optional attribute that represents the sequence label associated with this **List_Item** instance with a maximum multiplicity of one
valid_value	**Text_Element**	0..1	An optional attribute which specifies the value entered into the response when this list item is selected if that value is not the value_meaning with a maximum multiplicity of one

Reference	Class	Multiplicity	Description	Inverse	Precedence
item_container	**List_Field**	1..1	The **List_Field** to which the **List_Item** belongs where the maximum multiplicity is one	list_item	no
dependent_element	**Form_Design_Element**	0..*	An optional association to a set of **Form_Design_Element** instances whose availability is determined by the selection of the list item represented by this instance where the maximum multiplicity is unbounded	list_item_dependency	yes
default_item_of	**List_Field**	0..1	An optional association indicating that this **List_Item** instance should be offered as the default for the associated **List_Field** with a maximum multiplicity of one	default_item	no

5.5.17 List_Item_Selected_State

List_Item_Selected_State is an association metaclass, each instance of which represents the availability state of a dependent_element when the associated **List_Item** is selected.

Superclass

None

Attribute	DataType	Multiplicity	Description
list_item_selected	**Target_Element_State**	1..1	A Boolean flag which indicates if the dependent_element is available for completion when the **List_Item** that it depends upon it is selected. For example, consider a question “are you married?” with valid values of “yes” and “no”, where the valid value “yes” guards the follow-on question “what was your maiden name?”. The list_item_selected attribute should have the value “true” to ensure that selecting “yes” allows the follow on question to be completed.

5.5.18 Rule

Rule is a metaclass whose instances are groups of binary operations that capture functional dependencies between **Form_Design_Element** instances. **Rule**s provide detailed and flexible explanations of the state, value and appearance of the represented form both when initially presented to the user for completion and during the entry of data.

Superclass

None

Attribute	DataType	Multiplicity	Description
name	String	1..1	The signifier associated with the **Rule**

Reference	Class	Multiplicity	Description	Inverse	Precedence
condition	**Expression**	0..*	An optional association between a set of binary operation **Expression** instances which defines the conditions where the **Rule** applies with a maximum multiplicity of unbounded	containing_rule	yes
consequence	**Expression**	1..*	An association between a set of binary operation **Expression** instances which define consequences that should be satisfied where the **Rule** applies with a maximum multiplicity of unbounded	containing_rule	yes

5.5.19 Constant

Constant is a metaclass that supports the declaration of a constant in a local expression within the scope of a **Form_Design_Element**.

Superclass

Expression

Attribute	DataType	Multiplicity	Description
value	String	1..1	The value of the constant with a maximum multiplicity of one

5.5.20 Expression

Expression is an abstract metaclass which allows the recording of expressions that associate an ordered set of **Constant**, **Variable** and **Operation** instances. For example, condition{person.ecog IN (0,1,2)}.

Superclass

None

Reference	Class	Multiplicity	Description	Inverse	Precedence
containing_rule	**Rule**	1..1	The association specializing a particular **Rule** as the consequence **Expression**, where the maximum multiplicity is one	consequence	no
containing_rule	**Rule**	1..1	The association specializing a particular **Rule** as the condition **Expression**, where the maximum multiplicity is one	condition	no
containing_operation	**Operation**	0..1	The optional, ordered association containing the operations that are part of an **Expression**, where the maximum multiplicity is one	expression	no

5.5.21 Variable

Variable is an abstract metaclass that supports the declaration of a named variable in a local expression within the scope of a **Form_Design_Element**.

NOTE Named variables store the values of responses and the states of form design elements during the completion of an instance of a **Form_Design** ensuring that rules are succinct and readable.

Superclass

Expression

Attribute	DataType	Multiplicity	Description
identifier	String	1..1	The identifier of the **Form_Design_Element** that is the source of the variable with a maximum multiplicity of one
property	Property	1..1	The property of the identified **Form_Design_Element** that provides the variable value with a maximum multiplicity of one

5.5.22 Operation

Operation is an abstract metaclass whose valid instances are binary operations used to compose condition and consequence expressions. These expressions can be used to construct dependencies between **Form_Design_Element** instances, e.g. **List_Item** and other **Form_Design_Element** instances.

Superclass

Expression

Attribute	DataType	Multiplicity	Description
operation	Operation_Type	1..1	The operation invoked in an **Expression** with a maximum multiplicity of one

Reference	Class	Multiplicity	Description	Inverse	Precedence
expression	Expression	1..2	The set of **Operation** instances associated by this **Expression** where the minimum multiplicity is maximum multiplicity is two	Containing_ operation	yes

5.5.23 Reference_Document

Reference_Document is a metaclass whose instances are documents that provide pertinent details for consultation about a subject.

Superclass

None

Attribute	DataType	Multiplicity	Description
language	String	0..1	An optional attribute which identifies the language of the associated **Reference_Document** according to the encoding of ISO 639-3, where the maximum multiplicity is one
title	String	0..1	An optional attribute which is the title of the reference document, or some name by which it may be readily identified **where the maximum multiplicity is one**
uri	String	0..1	An optional attribute which is the URI by which the reference document may be accessed **where the maximum multiplicity is one**
document	BLOB	0..1	An optional attribute, which is the reference document itself, where the maximum multiplicity is one

NOTE It is expected that one of title, uri and document will be present in implementations of this document.

This class has the same purpose as, and may be replaced with the **Reference_Document** class in MDR Metamodel where both standards are implemented.

5.5.24 Datatype

Datatype is a metaclass whose instances each represent a set of distinct values, characterized by properties of, and by operations on those values.

Superclass

None

Attribute	DataType	Multiplicity	Description
name	String	1..1	The designation of the **Datatype**
description	String	0..1	An optional description of the **Datatype**
scheme_ reference	String	1..1	A reference identifying the source of the **Datatype** specification

This class has the same purpose as, and may be replaced with the Datatype class in MDR Metamodel where both standards are implemented.

5.5.25 Unit_Of_Measure

Unit_Of_Measure is a metaclass, each instance of which models a unit of measure, the units in which the associated answers on the form are measured.

Superclass

None

Attribute	DataType	Multiplicity	Description
unit_text	String	1..1	The designation of the **Unit_Of_Measure**

This class has the same purpose as, and may be replaced with the Unit_of_Measure class in MDR Metamodel where both standards are implemented.

Annex A
(normative)

MDR Mapping Package

A.1 General

NOTE The MDR Mapping Package will be generalized and moved into ISO/IEC 19763-10.

A frequent requirement in the registration of form designs is to relate questions to metadata elements stored in an MDR registry. This class supports the creation of typed associations between form design elements and metadata elements in an MDR metamodel metadata registry, further illuminating the meaning of questions in a registered form design.

Generally, any form design element may participate in a typed association with any identified item in a MDR metadata registry through the **Form_Design_Element_Association** class, but specializations of this association are provided to specifically associate questions with data elements and sections with concepts.

Where the MDR registry acts as a question bank used in the design of Form Designs, MDR_Association_Type may be extended to reflect the process of derivation provided that additional types are proper subtypes of the enumerations already provided.

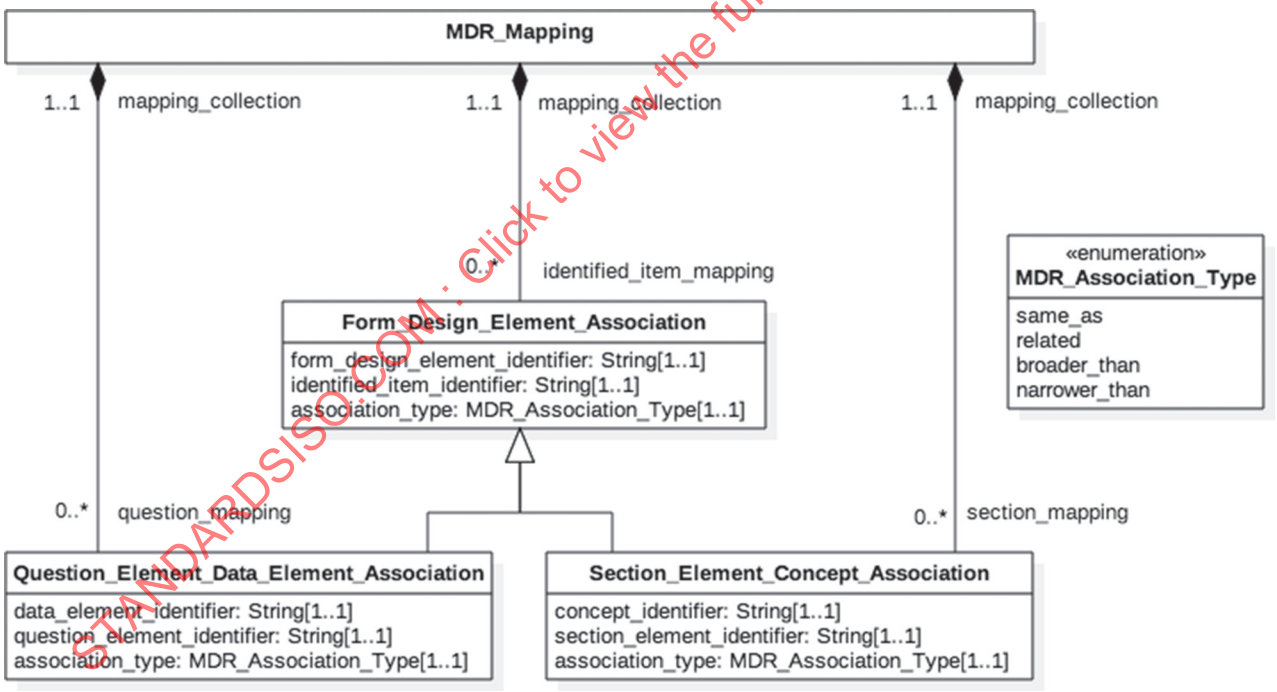

Figure A.1 — MDR Mapping Package

A.2 Basic Types and Enumerations in MDR Mapping Package

A.2.1 MDR_Association_Type

MDR_Association_Type is an enumeration of values for describing the type of association between items from a compliant metadata registry and a **Form_Design**.

Datatype

String

Metaclass	**Attribute**
Question_Element_Data_Element_Association	association_type
Section_Element_Concept_Association	association_type

Value	**Description**
same_as	Indicates that the entry in the MDR and the **Form_Design_Element** match exactly and there is no narrowing or broadening of meaning or representation. NOTE 1 For example, in the specific case of a mapping between a MDR metamodel **Data_Element** and a **Question_Element**, this would mean that the question text in the context of the whole form has exactly the same meaning as the definition of the **Data_Element** and the **Response** type and attributes match the **Value_Domain** exactly.
related	Indicates that the MDR element and the **Form_Design_Element** share some general, unspecified conceptual relationship. See skos:related. NOTE 2 Implementations may wish to subdivide this relationship category to explain cases where valid values are omitted or added to the list item collection, where formatting rules are tightened or relaxed or where data types or units of measure are changed without significantly affecting the idea conveyed in the data element/question pairing.
broader_than	Indicates that the **Form_Design_Element** has a broader meaning than the MDR element. NOTE 3 For example, in the specific case of a mapping between a MDR metamodel **Data_Element** and a **Question_Element**, this would mean that the question text meaning in the context of the form completely encompasses the definition of the **Data_Element** while the **Response** type and attributes match the **Value_Domain** exactly.
narrower_than	Indicates that the **Form_Design_Element** has a narrower meaning than the MDR element. NOTE 4 For example, in the specific case of a mapping between a MDR metamodel **Data_Element** and a **Question_Element**, this would mean that the question text has the same meaning as the definition of the **Data_Element**, the **Response** type and attributes match the **Value_Domain** exactly, but the context of the whole form narrows the meaning of the **Data_Element**.

A.3 Metaclasses in MDR Mapping

A.3.1 MDR_Mapping

MDR_Mapping is a metaclass whose valid instances provide a means by which to map between instances of **Form_Design_Elements**. For example, to map a **Question_Element** a particular data element, either from MDR Registry, or some other data element specification, for the purpose of defining the semantics and input constraints of the **Question_Element** and its **Response**s. See Figure 6.

Superclass

None

Attribute	DataType	Multiplicity	Description
None			

Reference	Class	Multiplicity	Description	Inverse	Precedence
question_mapping	**Question_Element_Data_Element_Association**	0..*	The association defining the set of instances of Question_Element_Data_Element_Associations for Question_Elements in the Form_Design	mapping_collection	yes
section_mapping	**Section_Element_Concept_Association**	0..*	The association defining the set of instances of **Section_Element_Concept_Associations** for **Section_Element** instances in the **Form_Design**	mapping_collection	yes
identified_item_mapping	**Form_Design_Element_Association**	0..*	The association defining the set of instances of **Form_Design_Element_Association** for **Form_Design_Element** instances in the **Form_Design**	mapping_collection	yes

A.3.2 Form_Design_Element_Association

Form_Design_Element_Association is a metaclass whose valid instances map a **Form_Design_Element** to any identified item in a metadata registry, for example, an MDR Registry. This would allow for a **Media_Element** to be associated to an identified concept, or for less well-typed associations, between questions and concepts.

Superclass

None

Attribute	DataType	Multiplicity	Description
form_design_element_identifier	String	1..1	The unique identifier of the particular **Form_Design_Element** in this **Form_Design** mapped to a specific **Identified_Item** through this instance of the **Form_Design_Element_Association**, where the maximum multiplicity is one
identified_item_identifier	String	1..1	The unique identifier of a specific **Identified_Item** mapped to a particular **Form_Design_Element** in this **Form_Design** through this instance of the **Form_Design_Element_Association**, where the maximum multiplicity is one
association_type	**MDR_Association_Type**	1..*	Categories describing the association. The attribute is enumerated using **MDR_Association_Type**.

Reference	Class	Multiplicity	Description	Inverse	Precedence
mapping_collection	**MDR_Mapping**	1..1	The association of this instance of **Form_Design_Element_Association** with a particular instance of **MDR_Mapping** where the maximum multiplicity is one	identified_item_mapping	no

A.3.3 Question_Element_Data_Element_Association

Question_Element_Data_Element_Association is a metaclass whose valid instances map a **Question_Element** to a data element in a metadata registry, for example, an MDR Registry.

Superclass

Form_Design_Element_Association

Attribute	DataType	Multiplicity	Description
data_element_identifier	String	1..1	The unique identifier of a specific registered data element mapped to a specific **Question_Element** in this **Form_Design** where the maximum multiplicity is one
question_element_identifier	String	1..1	The unique identifier of the particular **Question_Element** in the **Form_Design** mapped to the data element through this instance of the **Question_Element_Data_Element_Association**, where the maximum multiplicity is one
association_type	MDR_Association_Type	1..*	Categories describing the association. The attribute is enumerated using MDR_Association_Type.

Reference	Class	Multiplicity	Description	Inverse	Precedence
mapping_ collection	MDR_ Mapping	1..1	The association of this instance of **Question_ Element_Data_ Element_Association** with a particular instance of **MDR_Mapping** where the maximum multiplicity is one	question_ mapping	no

A.3.4 Section_Element_Concept_Association

Section_Element_Concept_Association is a metaclass whose valid instances map a **Section_Element** to a concept in a metadata registry, for example, a metadata registry conforming to MDR metamodel ISO/IEC 11179-3, or to a concept in a terminology. This may be used to record the narrowing of the meaning of generic data elements that have been used on the form, or a data element that is a composition of multiple data elements making up a standard section.

Superclass

Form_Design_Element_Association

Attribute	DataType	Multiplicity	Description
concept_ identifier	String	1..1	The unique identifier of the specific concept mapped to a **Section_Element** in this **Form_Design** where the maximum multiplicity is one
section_ element_ identifier	String	1..1	The unique identifier of the particular **Section_ Element** in the **Form_Design** mapped to the concept, where the maximum multiplicity is one
association_type	MDR_ Association_ Type	1..*	Categories describing the association. The attribute is enumerated using MDR_Association_Type.

Reference	Class	Multiplicity	Description	Inverse	Precedence
mapping_ collection	MDR_ Mapping	1..1	The association of this instance of **Section_ Element_Concept_ Association** with a particular instance of **MDR_Mapping** where the maximum multiplicity is one	section_ mapping	

Annex B
(informative)

Description of the metamodel

B.1 General

The metamodel for form design registration is intended to provide facilities for recording the logical design of a form and associating it with, in particular, data standards (MDR registry), other form designs, other relevant datamodels (see ISO/IEC 19763-12) and logical (MFD metamodel, MDR metamodel) models. This provides structured documentation on the nature and design of an individual form, ensures that data captured according to the form might be better understood and that design decisions in the creation of the form design from other form design, data models or ER diagrams may be explained in support of data sharing Master Data Management and the Semantic Web.

B.2 Relationships between model levels

This document addresses four layers: the Form Metamodel, which is the subject of this document; the Form Design, which is a specification of a form according to this document; an instance of a form design that has been deployed into an information system and a data document that has been created through the completion of the form, which may or may not be accompanied by the form semantics, depending upon the method of data exchange — paper (full form semantics), XML (normally containment structure only), RDBMS transaction (data shredded into third normal form).

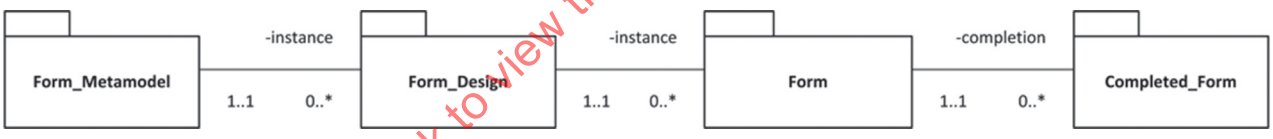

Figure B.1 — Modelling levels in this document

This document needs to address each of the three layers below it. It provides a language to standardize form design, it describes those aspects of form behavior that have semantic import and provides boundaries to the data space that a valid completed instance of the form can occupy. By taking control of each of these layers it provides for interoperability between data captured in different implementations, and between form designs that share common elements.

B.3 Structure

The structure of a form is modelled as an (optionally) ordered tree of **Form_Design_Element** instances inside a Form container. **Form_Design_Elements** may be **Section_Elements** that logically group other form elements, **Presentation_Elements** such as textual regions, pictures and audio clips, and **Question_Elements** that define how data is to be gathered by the form. A pair of optional **Section_Elements** with fixed semantics, the header and footer sections, are provided in this document although this function can be achieved without their use. Implementers may consider specializing a third instance of the **Section_Element** class for actual content, particularly when header and footer instances hold data fields required by an application framework such as a survey tool with an integral form designer.

Form_Design_Elements share optional name and style attributes. The name attribute allows an element to participate in a **Rule** but may be omitted when inheriting from the MDR metamodel types **Identified_Item** and **Designatable_Item**, the style attribute allows the registration of a stylesheet or other description of the layout of the components of a **Form_Element**, for example, the style attribute on

a **List_Field** might select between radio-buttons, a single select list box or a drop down combo-box in the presentation of the element. It may also indicate the position of the element on the rendered form design.

It is not generally intended that a **Question_Element** should contain a **Section_Element** although **Presentation_Element** instances often will, e.g. a box.

An annotated example of a form design indicating structural elements is given in Figure B.2.

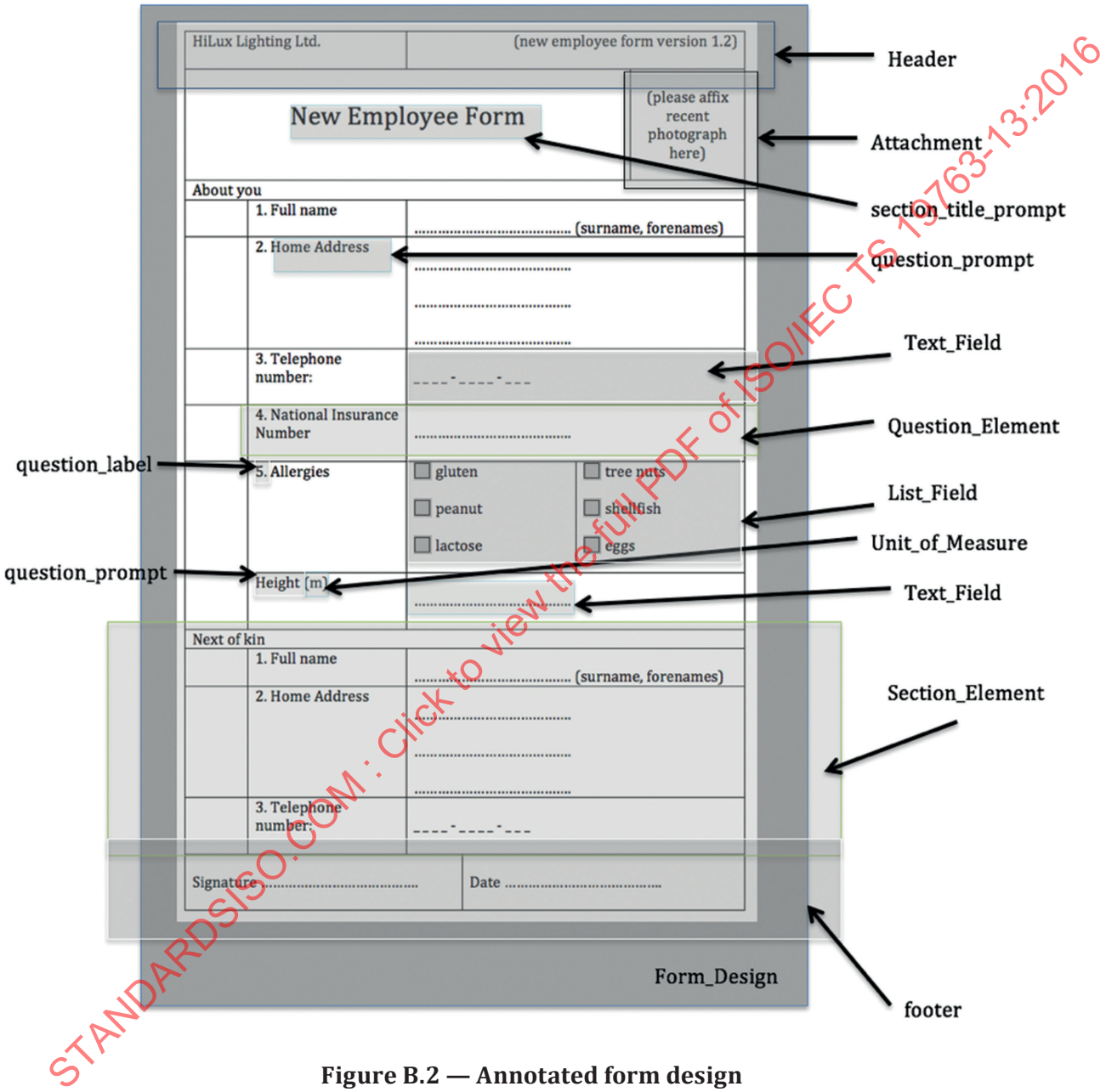

Figure B.2 — Annotated form design

B.4 Ordering

Form_Design_Elements may be ordered or un-ordered. Ordering is declared within a **Section_Element** and should always be selected when modelling or defining form designs that have functional dependencies between questions, overt semantic links, or where questions are used to skip, hide or indicate the relevance of following sections or questions. Common examples of constructs that require ordering include

Amplification, explanation or extemporization

Q1.prompt: what is your favourite colour?; Q2.prompt: why?

Q1.prompt: Does your partner smoke?; Q2.prompt: does this bother you?

Form workflow

Section 2 Q1.prompt: is the study subject alive? (yes/no) Following text element: "if the answer is *yes* then please go to section 3".

This is may be achieved either through a **Rule** (in pseudocode):

Form_Design_Element("S2Q1").rule(1).condition(1)=(.property(value)=yes)

Form_Design_Element("S2Q1").rule(1).consequence(1)=

(**Form_Design_Element**("S2Q2").property(visible)=false)

or through the use of a dependency relationship:

List_Item("S2Q1yes").dependent_element[**List_Item_Selected_State**.state=hidden]=

Form_Design_Element("S2Q2")

Functional dependency

Q1.prompt: Total taxable income (pounds sterling);

Q2.prompt: Total tax paid (pounds sterling);

Form_Design_Element("total_tax_paid").rule(1).consequence(1)=(.property(value)) LT

Form_Design_Element("total_taxable_income").property(value)/2)

Order should be the normal navigation order of the form, frequently the normal reading order for the (human) language and script of the form. Ordering of question elements should be achieved through assignment of Question_Number prompts or through document order as these mechanisms are visible to the user of the form.

The particular implementation of ordering in registry software is left to the implementer: in Java, one may declare multi-valued attributes or classes as lists (e.g. List<Section> sections); in XML one may use natural document order; however, relational implementations will require an extra column to persist the order of records in a child table.

B.5 Containment and repetition

The cardinality attribute of the **Section_Element** class allows the modelling of tables: the table is a repeating **Section_Element** with **Question_Element**s for each column. Tabular presentation is effected by the style attribute.

B.6 Questions and responses

A form design has no answers, only questions, responses and constraints. Questions may have prompts, which hold the main semantics of the answer that is to be placed in the **Response**, question numbers and additional instructions. Other text that is collocated with a question but has no expected semantic content is a presentation element. **Response** may be a **String_Field** which allows the entry of numbers and strings, a **List_Field** which allows the user to select from a menu of **List_Items**, and a **Lookup_Field** which fetches the currently available valid values for an answer from a web service or a database view. A **List_Field** may have a "fill-in" field which allows the user to enter a value that is not amongst the set of List_Items specified. Each **List_Item** has a textual prompt and optional labels or numbers and additional_text, and it may also set a dependency against another **Form_Design_Element** depending upon its state matching the value of the guard attribute. Thus, a list item with a specified set of dependent_elements and a **List_Item_Selected_State** state attribute of "selected" will discourage selection of the dependent **Form_Design_Elements** if that value is selected as the answer to the question element. If the state attribute is set to "unselected" then the dependent elements will not be accessible (relevant) *unless* that answer is selected.

B.7 Rules

Unlike a relational database, in which data items have been meticulously ordered to eliminate functional dependencies on non-key fields, all but the simplest of form designs have numerous overt functional dependencies between form elements that have often been designed to ensure consistency or to guide form completion. Some dependencies guide the user to complete certain questions in preference to others; others constrain and define valid completions of the form. Electronic forms also optionally can set the order and precedence of these constraints, but the order of the application of the constraints, provided they are logically consistent, does not affect the semantics of the completed form, the point at which a field becomes unavailable should not affect the semantics of the answers, only the fact that that field should not be completed if a certain set of conditions were met according to the normal reading order of the questions on the form.

This document provides two constructs to encode such logic: the dependent_element association (see above) and the **Rule** class. The **Rule** class allows the form designer to register simple binary expressions about instances of the **Form_Design_Element** class to define or constrain values on a valid, completed form or the availability of questions on that form. A local expression consists of a *condition*, which should be satisfied for the expression to be in scope, and a *consequence* that applies when it is in scope.

The constraint language that is used to compose **Local_Expression** statements defines a small number of basic operations that shall be supported by the form implementation language. Essential operations direct the user to ignore certain **Form_Design_Element** instances (**Form_Design_Element**(*identifier*).available), constrain the values of **Form_Design_Element** instances (**Form_Design_Element**(*identifier*).value), set default values (**Form_Design_Element**(*identifier*).default_value). Other operations affecting the style of the form may be added as extensions, but there is no requirement in this document to support them. Similarly, the order and precedence of these operations could be set in an implementation of this document, but this is out of scope for this document itself.

While a **Form_Design** has no answers, its **Rules** need to address those answers supplied by the user as the form instance is completed. Thus, the **Rule** has these additional properties that are not enumerated in the form design model itself so that runtime behavior can be declared alongside the model.

B.8 Representation

By default, a *label* on a form is of type **Sign** which allows the representation of a concept in textual, pictorial or other modality. This document supports the association between **Text_Element** instances and **Media_Element** instances by an explicit *representation-meaning* association so that logos and company names or slider scales and legends can be declared where the meaning of the image, moving picture or audio file is explained by the **Text_Element** instance. Thus, alternate versions of the form with pictorial and audio content may be presented for the poorly literate. Direct linkage to concepts may be achieved through the MDR Mapping functionality.

B.8.1 Rule Example

When the Question "Dead" is true, there are three consequences: do not permit an answer to the Question "Performance Status" and set its style to (CSS) visibility:hidden, the "cause-of-death" question should be asked, and the value for the "date-of-death" question should be greater than or equal to the question "last-date-seen-alive".

Represented as pseudo-language statements:

Rule "Form behavior and validation rules when Dead is true":

condition

Form_Design_Element("dead").property(value)=true

consequences

Form_Design_Element ("performance-status").property(available)=false

Form_Design_Element ("performance-status").property(style)=visibility:hidden

Form_Design_Element ("cause-of-death").property(available)=true

Form_Design_Element ("date-of-death").property(value)

> Form_Design_Element ("last-date-seen-alive").property(value)

The representation in MFI Form design registration is as follows.

Class	Attribute	Value
Rule	name	Form behaviour and validation rules when Dead is true
Condition		
Variable	form_design_element_identifier	dead
	property	value
Operation	operation	EQ
Constant	value	true
Consequence		
Variable	form_design_element_identifier	performance-status
	poperty	available
Operation	operation	EQ
Constant	value	false
Consequence		
Variable	form_design_element_identifier	performance-status
	property	style
Operation	operation	EQ
Constant	value	visibility:hidden
Consequence		
Variable	form_design_element_identifier	cause-of-death
	poperty	available
Operation	operation	EQ
Constant	value	false
Consequence		
Variable	**form_design_element**_identifier	date-of-death

Class	Attribute	Value
	property	value
Operation	operation	GT
Variable	**form_design_element**_identifier	late-date-seen-alive
	property	value

Since this document relies upon standard mathematical constructions, a complete enumeration of the types of binary operations that should be supported is not given. However, the list provided will be supported by all compliant implementations.

B.9 Templates

The cardinality of the relationships is designed to allow the creation of empty **Section_Element** instances. One use of this is the definition of a **Form_Design_Template**, where an empty section is declared for content extension: in a clinical follow up form template, standard content would be declared to ensure that censor dates are uniformly collected, and an empty section provided for content specific to the disease; in an application framework for the delivery of form designs, standard fields required by the application framework could be declared in a template so that application metadata can be explained and persisted to users who later receive form data.

B.10 Reference documents

Frequently, form designs need significant supporting documentation, including specifications for their correct use, reference implementations, and copyright statements. No specific types of reference document are normative in this document — facilities for typing reference documents are provided in the MDR Metamodel.

Annex C
(informative)

Relationship of metaclasses to the MDR Metamodel

C.1 Summary

As explained in ISO/IEC 19763-10, instances of the metaclasses defined in this document may be extended by the types defined in the MDR Metamodel as follows.

— **Form_Design** may be extended as an **Identified_Item**, **Designatable_Item**, **Registered_Item**, **Administered_Item** and **Classifiable_Item**.

— **Form_Design_Element** may be extended as an **Identified_Item**, **Designatable_Item** and **Classifiable_Item**.

— Any instance of a **Form_Design_Element** may be mapped to an instance of a **Concept**.

— Any instance of a **Question_Element** may be mapped to an instance of a **Data_Element**.

— **List_Item** may be extended as an **Identified_Item**, any instance of which may be mapped to a **Concept** and/or **Permissible_Value**.

— **Rule** may be extended as an **Identified_Item** and **Designatable_Item**.

C.2 Explanation

MDR metamodel has two major functions in relation to this standard. It provides:

— a rich meta-data and administrative model that can be used to identify, name, register, administer, explain and classify **Form_Design** instances, **Form_Design_Element** instances and many of the other classes described in the main body of this document. This provides essential metadata for the registration of form designs and supports the functions of a form registry;

— a detailed model of **Question_Element** metadata so as to better describe the meaning of the questions and responses, and to map **List_Item** instances through valid values or value meanings to terminologies and ontologies.

For instance, the *MDR metamodel Identified_Item* class provides capabilities to create a set of globally unique identifiers, allowing one to identify a **Form_Design** instance in the local collection and also maintain suitable identifiers relating to its inclusion in other collections. Going further, one can associate a **Form_Design** instance with an *MDR metamodel Administration_Record* which adds a set of metadata slots suited to the management of the development and approval for **Form_Design** instances in an information environment controlled by a master data manager. If a **Text_Element** is a MDR metamodel *Designatable_Item*, one can immediately register multilingual form designs.

NOTE While it is possible to translate some form designs at an individual question/section level and find the collection of these individually translated elements is easily comprehendible in the alternate languages, an alternative is to associate and map two separately registered form designs that have been wholly translated into other languages.

Finally, to facilitate the retrieval of form designs by purpose or domain of interest, one might also make a **Form_Design** instance of a MDR metamodel *Classifiable_Item* so that it might be associated with 0..* MDR metamodel *Classification_Scheme_Items*. In this way, one could broadly associate form designs that have a similar purpose — form designs in an administration that collect demographic information, or