	Technical Model Infrastructure Specification Part 4: Technical Model Representation, Constraint and Serialization			
	Programme	NPFIT	Document Record ID Key	
	Sub-Prog / Project	Informatics Data Standards Programme	NPFIT-FNT-TO-DPM-1020.03	
	Prog. Director	D. Perry	Status	WORKING DRAFT
	Owner	L. Sato	Version	0.3
	Author	J. Arnett	Version Date	2010-03-29

Logical Record Architecture for Health and Social Care:

Technical Model Infrastructure Specification Part 4: Technical Model Representation, Constraint and Serialization

Amendment History:

Version	Date	Amendment History
0.1	2010-02-02	Incomplete first draft for comment
0.2	2010-02-15	<p>Entire document</p> <ul style="list-style-type: none"> Changed each and every occurrence of 'EN 13606-1:2007' to 'BS EN ISO 13606-1:2008'. Changed each and every occurrence of 'ISO 21090' to 'ISO/FDIS 21090'. <p>Section 2.1 Reference Models</p> <ul style="list-style-type: none"> Para 4 – explained origin of Participations model. Added infrastructure specification citations. <p>Section 3.1.3 Property Feature</p> <ul style="list-style-type: none"> Para 6 - removed 'This is optional and in addition to the mandated use of OCL to express fixed values.'
0.3	2010-03-29	<p>Updated document title to conform to others in infrastructure series.</p> <p>Entire document</p> <ul style="list-style-type: none"> Removed all references to undefined and potentially ambiguous expression 'domain-of-use'. <p>Section 2 LRA Technical Model Types</p> <ul style="list-style-type: none"> Updated to clarify meaning and purpose of Technical Model types in general. <p>Section 2.2 Model Reuse Patterns</p> <ul style="list-style-type: none"> Added para 2 to clarify purpose of model reuse patterns. Updated para 3 to clarify rules concerning addition of 'new' data types. <p>Section 2.3 Domain Models</p> <ul style="list-style-type: none"> Updated to clarify meaning and purpose of the Domain Models in particular. <p>Section 2.5 Comparison of Model Artefact Types</p> <ul style="list-style-type: none"> Added list of the LRA Technical Model types and the corresponding H7 v3 and <i>openEHR</i> model types that share the same or similar levels of abstraction.

Forecast Changes:

Anticipated Change	When

Reviewers:

This document must be reviewed by the following:<author to indicate reviewers>

Name	Signature	Title / Responsibility	Date	Version
------	-----------	------------------------	------	---------

S. Bentley				
N. Oughtibridge				

Approvals:

This document must be approved by the following: <author to indicate approvers>

Name	Signature	Title / Responsibility	Date	Version
S. Bentley	Acting Head, Logical Record Architecture			
K. Lunn	Director, NHS Data Standards & Products			

Distribution:

<Author to say who the document will be distributed to>

Document Status:

This is a controlled document.

Whilst this document may be printed, the electronic version maintained in FileCM is the controlled copy. Any printed copies of the document are not controlled.

Related Documents:

These documents will provide additional information.

Ref no	Doc Reference Number	Title	Version
1.	NPFIT-SHR-QMS-PRP-0015	Glossary of Terms Consolidated.doc	
2.	NPFIT-FNT-TO-DPM-0911.12	LRA Artefacts Overview	1.2
3.	NPFIT-FNT-TO-DPM-0935.01	LRA Technical Model Infrastructure Specification Part 1: Care Components	0.5
4.	NPFIT-FNT-TO-DPM-0922	LRA Technical Model Infrastructure Specification Part 2: Participations	0.6
5.	Available from http://www.omg.org/	OMG Unified Modeling Language (OMG UML), Superstructure	2.1.2
6.	Available from http://www.hl7.org	HL7 Version 3 Standard, January 2010	
7.	Available from http://www.omg.org/	UML 2.0 OCL Specification	
8.		LRA Terminology Binding Technical Specification	0.6
9.	Available from http://www.omg.org/	OMG Unified Modeling Language (OMG UML), Infrastructure	2.1.2
10.	Available from http://www.omg.org/	Meta Object Facility (MOF) Core Specification	2.0
11.	NPFIT-FNT-TO-DPM-0936.01	LRA Data Types Specification	0.3

12.		ISO/IEC 19503:2005 Information technology - XML Metadata Interchange (XMI)	
13.	Available from http://www.omg.org/	MOF 2.0/XMI Mapping	2.1.1
14.	NPFIT-FNT-TO-DPM-0940.01	LRA Technical Model Tooling, Serialisation and Interfacing	
15.		ISO/FDIS 21090:2007 Health informatics – Harmonized data types for information interchange	
16.		BS EN ISO 13606-1:2008 Health informatics — Electronic health record communication — Part 1: Reference model,	
17.		ISO/IEC 10746-1:1998 Information technology - Open Distributed Processing - Reference model: Overview	

Glossary of Terms:

List any new terms created in this document. Mail the NPO Quality Manager to have these included in the master glossary above [1].

Term	Acronym	Definition
Constraint		Any model construct that asserts a restriction upon the semantics of one or more other model constructs
Meaning based retrieval		Retrieval involving selection and processing of parts of a care record (or a set of care records) based on the formally expressed meaning of the contained items of information and of the links between them.
Metamodel		A formal specification of the content, structure, function and / or behaviour of conforming model instances.
Model		A formal specification of the content, structure, function and / or behaviour of conforming data instances.
Domain (of discourse)		A particular area or field of interest defined by the things that comprise it.
Domain of the LRA		A domain of discourse comprising those aspects of health and social care within England about which information is routinely recorded and shared.

Contents

1	About this Document	7
1.1	Purpose.....	7
1.2	Audience	7
1.3	Content.....	7
2	LRA Technical Model Types.....	8
2.1	Reference Models	10
2.2	Model Reuse Patterns.....	10
2.3	Domain Models	12
2.3.1	ENTRY Domain Models.....	13
2.3.2	ELEMENT Domain Models	13
2.3.3	PARTICIPATION Domain Models	13
2.4	Constrained Domain Models	13
2.4.1	COMPOSITION Model.....	14
2.4.2	ENTRY Constrained Domain Models	14
2.4.3	ELEMENT Constrained Domain Models.....	14
2.5	Comparison of Model Types	14
3	LRA Technical Model Representation	15
3.1	Core Constructs	17
3.1.1	Package.....	18
3.1.2	Class.....	18
3.1.3	Property Feature	19
3.1.4	Constraint Feature	19
3.1.5	Relationships	20
3.2	LRA Technical Model UML Representation	25
3.2.1	Stereotypes.....	26

3.2.2	Tagged Value Properties	29
3.2.3	Profile Constraints	30
3.3	LRA Technical Model Standard Representation	30
4	LRA Technical Model Constraint	31
4.1	Class and Data Type Level Constraints	31
4.1.1	Restriction of Intension	31
4.1.2	Class and Data Type Specialisation	32
4.1.3	Reference Model Class Realization	32
4.1.4	Data Type Flavors	33
4.1.5	Ad hoc Constraints	33
4.2	Property Level Constraints	34
4.2.1	Restriction of Attribute Function	34
4.2.2	Constraining Property Type	34
4.2.3	Constraining Occurrence	35
4.2.4	Asserting Fixed Values	37
4.2.5	Constraining Navigability	37
4.3	Value Space Constraints	38
4.3.1	URF for Value Space Constraints	42
4.4	LRA Technical Model Derivation	43
5	LRA Technical Model Serialization	44
5.1	Data, Models and Metamodels	44
5.2	SRF Metamodel Requirements and Design Rationale	44
6	Appendices	47
A.1	LRA Core Modelling Constructs	47

1 About this Document

1.1 Purpose

This document specifies the formal means of representation, constraint and serialization of LRA Technical Models¹.

1.2 Audience

The intended audience of this specification is any individual, group or organisation involved in the development or use of the *LRA*.

1.3 Content

This document comprises the following sections:

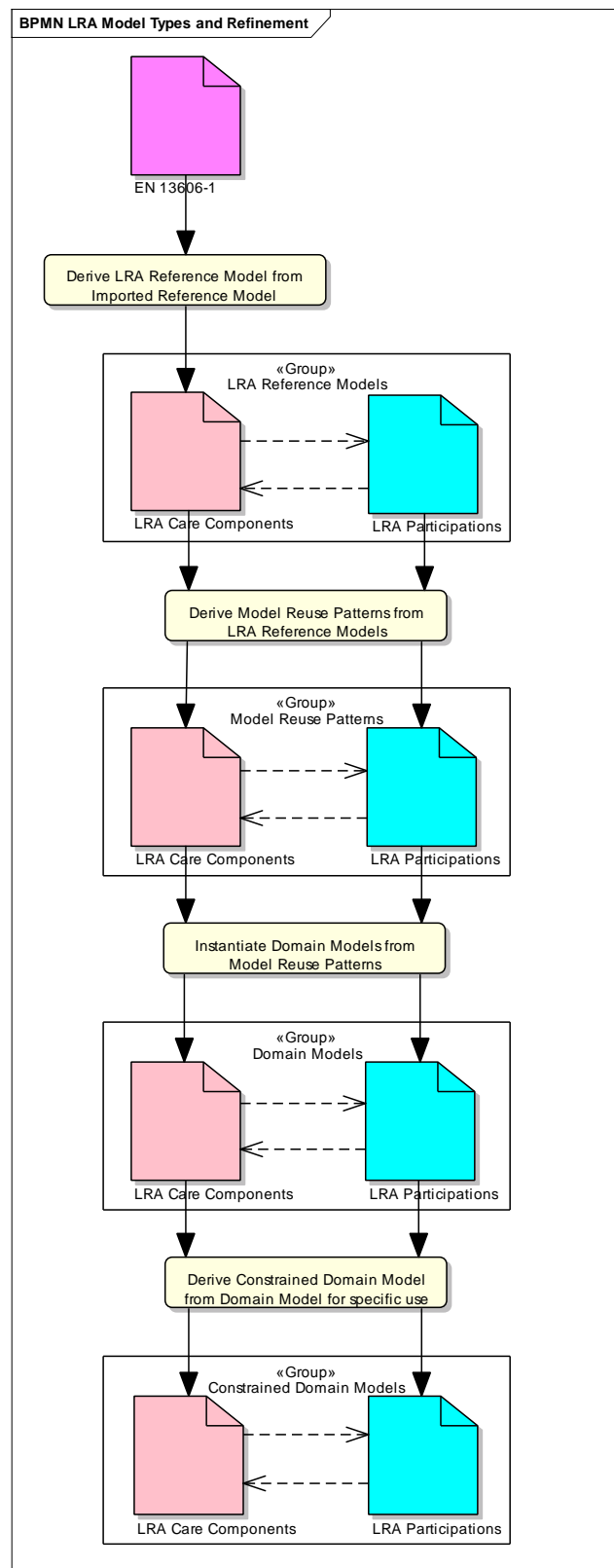
- LRA Technical Model Types - describes the different Technical Model types specified by the LRA PE (Production Environment), the purpose of each and their relationships to each other.
- LRA Technical Model Representation - describes the two representational forms used for representing LRA Technical Models, the core UML constructs they share and the extensions specific to each form.
- LRA Technical Model Constraint - describes the various types of constraint that may be applied to LRA Technical Models together with their methods of specification. This section also lists the various constraint types used (or available to use) to derive the successive instances of each Technical Model type.
- LRA Technical Model Serialization – this section species the formal metamodel and schema for serializing Technical Model type instances. **The section remains to be completed at the time of publication of this draft version of the document.**

¹ Within this document the term Technical Model refers to an instance of a LRA Technical Model type as described in section 2.

2 LRA Technical Model Types

LRA Technical Models can be classified into four types based on their role within the Production Environment (PE). Imported and LRA-specific *Reference Models* define controlled and finite sets of classes, relationships and instance constraints used to construct *Domain Models* using instantiable *Model Reuse Patterns*. Finally *Constrained Domain Models* are created by constraining general domain models for use within one or more domains.

Figure 1 illustrates the different model types within the PE and shows the refinement processes by which they are derived from one another. The subsections that follow describe the various model types in more detail.

**Figure 1 LRA PE Model Types**

2.1 Reference Models

Reference Models form part of the PE infrastructure and serve to formalise the construction and interpretation of Domain Models by defining the basic structures, semantics and constraints to which Domain Models must conform.

All LRA Reference Model classes are defined in a package hierarchy under a `Ira` root package. The hierarchy includes Reference Models for:

- Data Types [11], specified under `Ira::datatypes`;
- Care Components [2], specified under `Ira::technical::en13606`; and
- Participations [3], specified under `Ira::technical::participations`.

The Data Types and Care Components Reference Models are derived by constraint from the imported Reference Models ISO/FDIS 21090 [15] and BS EN ISO 13606-1:2008 [16] respectively. The Participations reference model is based largely on a reconciliation of multiple models present in MIM 7.2.02 for describing the participation of roles and entities within the health record.

2.2 Model Reuse Patterns

Model Reuse Patterns are Reference Model abstractions and are often embedded within modelling tools as model element (e.g. class) templates for instantiating new Domain Model elements.

One important function of Model Reuse Patterns is to collapse all of the properties and constraints of a Reference Model class hierarchy into a single model template class as exemplified in Figure 2. The `GENERAL_ACTIVITY_ELEMENT` is an abstraction that incorporates all of the properties and constraints of the generalisation hierarchy of the `GENERAL_ACTIVITY_ELEMENT` Reference Model class, on the right of the diagram, into a single instantiable model element.

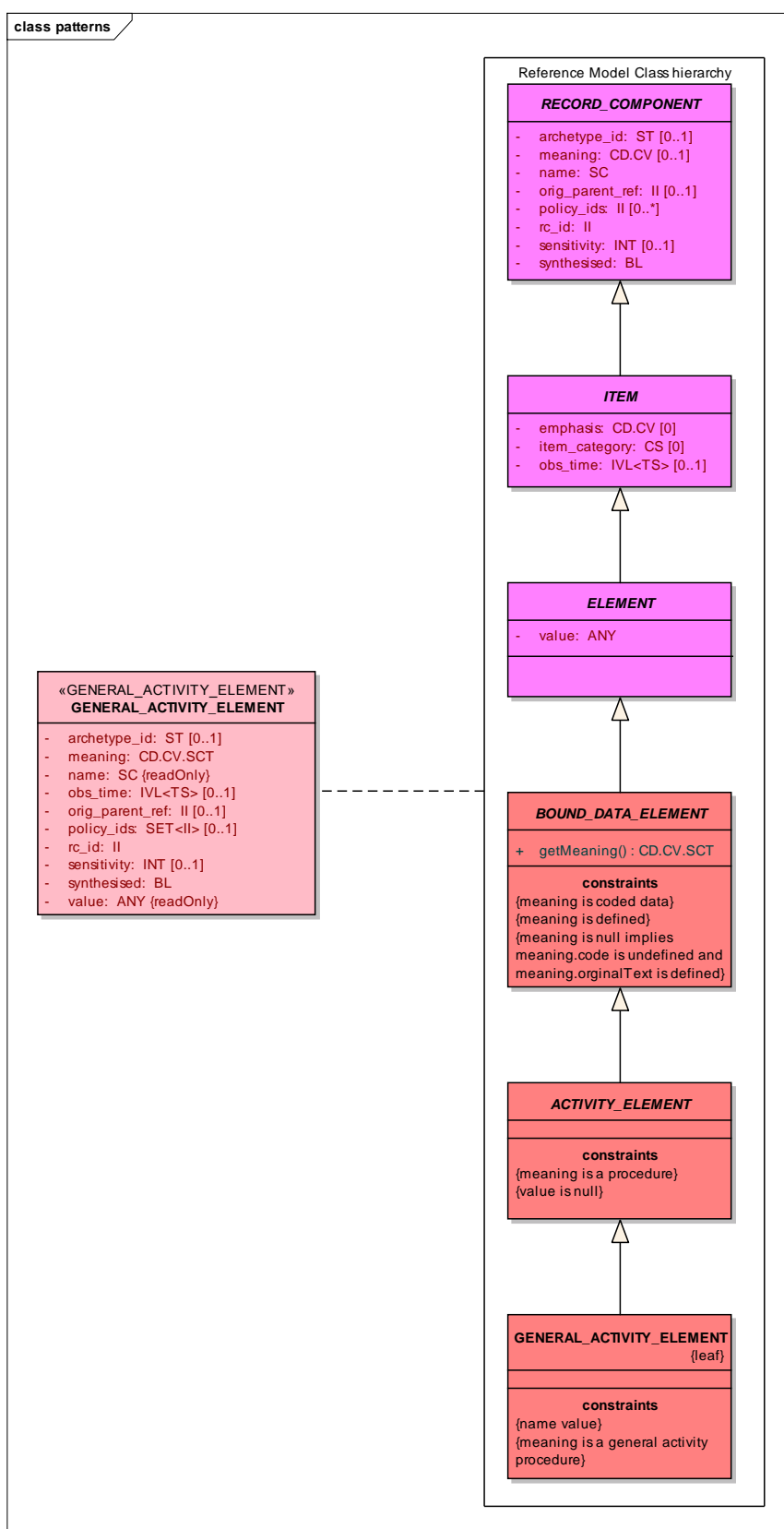


Figure 2 Model Reuse Pattern **GENERAL_ACTIVITY_ELEMENT** and its Reference Model generalisation hierarchy

Within the LRA Model Reuse Patterns are used to instantiate Care Components and Participations model elements which have embedded within them Data Type instances in the form of model element attributes. Data Types are therefore not instantiated as standalone model elements within LRA Domain Models. The LRA Data Types Reference Model [11] specifies a subset of the ISO/FDIS 21090 [15] data types. Data types defined by the general standard that are not currently members of the LRA subset MAY, however, be added to the LRA if or when use cases are encountered that require their inclusion and in turn be made available as attribute types within Model Reuse Patterns and / or Domain Models.

2.3 Domain Models

A Domain Model is a *domain*-level realization of one or more Reference Model classes. Informally, a domain (or domain of discourse) is a particular area, sphere or field of activity, thought, study or interest² defined (by implication) by the things that comprise it. The domain of the LRA comprises those aspects of health and social care within England about which information is routinely recorded and shared. More formally in predicate logic, a domain is a set of variables (or instances) to which logical quantifiers (expressed by words such as 'all', 'some' and 'no') may bind, e.g. '*all* subjects of care', '*some* procedure' or '*no* finding of...'.

It is important to distinguish the meanings of the term 'domain' as generally understood in data modelling and in formal logic from the occasional and narrower use of the term within healthcare informatics to mean a particular area of interest *within* healthcare. The terms *health care domains* or *clinical domains* (e.g. as used [16] - note the plural) imply this particular meaning and are synonymous in this respect. This is also *one* of the meanings defined for the term within the HL7 v3 Glossary. Special interest groups within HL7 such as Pharmacy, Laboratory or Patient Administration are known as domains and the messages that support their communication requirements are called DMIMs (Domain Message Information Models). Thus the term 'domain' within LRA technical modelling refers to its broader and more general use and NOT to the restricted meaning of a particular area of interest *within* healthcare.

To make sense of a domain (i.e. to be able to model it), abstraction is used to assert the defining qualities of the things of interest. These qualities may then be used to classify things into a manageable number of types. The qualities asserted by each type are inherited and can be applied to each and every subtype. Based on this notion, LRA Domain Models are generalised representations of the things of interest (and the relationships between them) within the LRA domain of discourse. Their purpose is to provide logical record structure definitions that are optimised for reuse in representing information consistently, independent of health or social care specialty, intervention setting, encounter type, episode type, recording practice and intended use. Domain Models are therefore in general underspecified in that they require constraining to satisfy precisely the information requirements of one or more

² Adapted from definition <http://www.thefreedictionary.com/domain> and other similar online definitions.

specified areas of interest. In terms of the ISO Open Distributed Processing Reference Model (RM-ODP) [17] a Domain Model corresponds to a *global* schema of the Information Viewpoint, i.e. a model that defines properties, constraints and behaviour that must be true for each and every instance of a (Domain Model) class.

Although Domain Models can be manually constructed (i.e. by copying the relevant parts of the Reference Model class hierarchy into a single element) it is more efficient and less error prone to instantiate them from predefined Model Reuse Patterns.

The LRA defines a number of Domain Model artefact types [3] for specific purposes as described in the following sub-sections.

2.3.1 ENTRY Domain Models

An ENTRY Domain Model is a domain-level realization of Reference Model class `Ira::technical::en13606::extract::ENTRY`. [3]

ENTRY Domain Models serve as containers for ELEMENT Domain Models. [2]

2.3.2 ELEMENT Domain Models

An ELEMENT Domain Model is a domain-level realization of one of:

- Reference Model class `Ira::technical::en13606::extract::UNBOUND_DATA_ELEMENT`; [3] or
- any concrete specialisation of Reference Model class `Ira::technical::en13606::extract::BOUND_DATA_ELEMENT`. [3]

ELEMENT Domain Models are independently reusable model structures that are designed to be further constrained to represent precisely the shared knowledge and technical semantics of one or more Candidate Data Element Definitions.

2.3.3 PARTICIPATION Domain Models

A PARTICIPATION Domain Model is a domain-level realization of one of:

- Reference Model class `Ira::technical::en13606::extract::RELATED_PARTY` or `Ira::technical::en13606::extract::FUNCTIONAL_ROLE`; [3]
- Reference Model class `Ira::technical::en13606::extended::CR_Participation`; [3] or
- any concrete Reference Model class defined within package `Ira::technical::participations`. [4]

PARTICIPATION Domain Models represent the details of the participating roles, and of the entities that play them, that may be associated with an ENTRY Domain Model.

2.4 Constrained Domain Models

Constrained Domain Models are Domain Models which have been constrained to satisfy precisely the information requirements of one or more particular areas of

interest. Constrained Domain Models are defined in a separate LRA Interface package as constrained specialisations of existing Domain Models. Any instance of a Constrained Domain Model element is therefore an indirect instance of a more general Domain Model class (i.e. generalised class). In terms of the ISO RM-ODP a Constrained Domain Model corresponds to a *local* schema of the Information Viewpoint, i.e. a model that defines the properties, constraints and behaviour that must be true for each and every instance of a (Domain Model) class in order to satisfy the information requirements of one or more specified areas of interest within the domain of the LRA.

The LRA defines a number of Constrained Domain Model artefact types [2] for specific purposes as described in the following sub-sections.

2.4.1 COMPOSITION Model

A COMPOSITION model is a domain-level realization of Reference Model class `Ira::technical::en13606::extract::COMPOSITION` intended for use within one or more specified areas of interest within the domain of the LRA.

A COMPOSITION Model serves as a logical container for the ENTRY Constrained Domain Models that relate to a notional single clinical encounter or documentation session. The clinical content so represented by COMPOSITION Model is intended to be committed within one EHR.

2.4.2 ENTRY Constrained Domain Models

An ENTRY Constrained Domain Model represents an ENTRY Domain Model constrained for use within one or more specified areas of interest within the domain of the LRA.

ENTRY Constrained Domain Models serve as containers for ELEMENT Constrained Domain Models. [2] An ENTRY Constrained Domain Model is designed to be used in its entirety and MAY be reused by multiple COMPOSITION Models to meet precisely a recurring requirement or set of requirements.

2.4.3 ELEMENT Constrained Domain Models

An ELEMENT Constrained Domain Model represents an ELEMENT Domain Model constrained for use within one or more specified areas of interest within the domain of the LRA.

ELEMENT Constrained Domain Models are used to represent the shared knowledge and technical semantics of Candidate Data Element Definitions and Element Domain Models. [2] An ELEMENT Constrained Domain Model is NOT intended be used independently of the ENTRY Constrained Domain Model within which it is defined.

2.5 Comparison of Model Types

Table 1 below presents a list of the LRA Technical Model types and the corresponding H7 v3 and *openEHR* model types that represent equivalent or similar levels of abstraction.

Table 1

LRA	HL7 v3	<i>openEHR</i>
Reference Model	HL7 v3 Reference Information Model	openEHR EHR Reference Models
Model Reuse Pattern	Model element available to drag and drop from HL7 v3 model design tools object palette	Archetyped element available to drag and drop from archetype design tool object palette
Domain Model	HL7 v3 Domain Message Information Model (DMIM) HL7 v3 NPfIT CDA Template model	Archetype
Constrained Domain Model	HL7 v3 NPfIT CDA Template model HL7 v3 Refined Message Information Model (RMIM)	Template

3 LRA Technical Model Representation

The LRA defines a specification for the logical representation of care record information. LRA Technical Models focus in particular on specifying logical record structures that support 'meaning-based retrieval', i.e. retrieval involving selection and processing of parts of a care record (or a set of care records) based on the meaning of the contained information and the links between them. The kinds of structure used by LRA Technical Models to represent care record information are defined as part of the Technical Model Infrastructure and are derived or adapted from a number of international standards including BS EN ISO 13606-1:2008, ISO/FDIS 21090 and SNOMED CT. This specification is concerned with the representation of the various kinds of structure used to represent care record information and not with the representation of the care record information per se. To this extent the document specifies a *meta*-representation. However, the term 'representation' is used here as its meaning is considered to be unambiguous within the context of this document and it is undoubtedly clearer to most readers.

The LRA supports two equivalent and interchangeable forms of Technical Model representation. These are called:

- UML Representational Form (URF)
- Standard Representational Form (SRF).

The two forms use a common set of core UML 2.1 class model constructs [9] but augment them in different ways to express LRA-specific features.

The URF is a UML based representation, with an underpinning UML XMI serialization, in which the core model constructs are extended using a UML profile. UML profiles provide a lightweight extension mechanism that enables UML models to be tailored to specific areas such as a particular technology, methodology or business domain. [5] The URF uses both standard and customised UML notational constructs and is designed to make models accessible to being viewed and edited by commercially available or open source UML tools. Domain Models and Constrained Domain Models are rendered in the form of UML class diagrams which although customised to express LRA semantics are composed mostly of standard notational constructs that anyone familiar with UML should be able to understand. The customised, LRA-specific notation is intended to make the models easier to understand. Reference Models and data types are represented formally using standard UML class diagrams. This is to help ensure that the underlying technical architecture is accurately represented using a software industry standard formalism and can be inspected without having to understand any of the LRA-specific notations used to represent Domain Models and Constrained Domain Models.

The underpinning UML XMI serialization of the URF is designed to facilitate model interchange between different UML tools. However, because of the different versions of UML XMI and more importantly differences in vendors' implementations (through the use of XMI's *extension* elements) lossless interchange of models between the tools of different vendors is not guaranteed. However, experience has shown that quite often, minor transforms can be applied to convert a source XMI implementation produced by one vendor's tool to a target implementation required for importing into another vendor's tool with little or no significant data loss.

Although the LRA has been developed in URF the formalism suffers from the following significant weaknesses.

1. The meaning of the various LRA-specific extensions to the UML, defined using UML profile constructs, resides outside of the URF. Thus despite being able to use standard commercial or open source UML tools to view, edit and exchange LRA models in URF, by definition such tools lack the LRA-specific functionality necessary to interpret the semantics of the various LRA-specific stereotypes, tags and constraint methods used.³
2. Most UML tools (so far as is known) do not support fully some key features of the LRA including linking of terminology bound attributes to their specified expression constraints, model documentation and redefinition of class attributes (see section 4.2). Tooling and methodology work rounds have therefore been developed to enable the URF to represent these features.

³ In fact the UML Superstructure Specification [5] states the target of the exchange of a model extended by a profile may not know the profile, and is not required to interpret a specific profile description. The destination environment interprets extensions only if it possesses the required profiles. In practice this is likely to be complicated by vendor-specific, and therefore inconsistent, processing of profile information.

3. For reasons of convenience the URF suffers from a lack of representational consistency. This is particularly the case with constraints in which some types of constraint are represented using the formal constraint feature provided by UML whereas other types of constraint are more conveniently represented using tagged values.
4. Exchange of LRA models in URF between different vendors' UML tools requires the UML XMI serialization to be translated from the vendor specific XMI implementation of the source tool to the vendor specific implementation of the target. Although the transformations needed may in some cases be minor, different implementations may in particular serialize profile features, which exist outside of the core UML model serialization, in very different ways. There is also a scalability issue to consider if there is an intention to support model interchange between all UML tools.
5. The XMI serialization of the URF is difficult to parse due to model features being split between the core UML model and the XMI extension.

As a result this document also specifies a SRF which is designed to address the inherent limitations of the URF. Furthermore, while the SRF is intended to provide a representational formalism tailored to meet the specific needs of developing and maintaining LRA Technical Models, it uses an international standard closely aligned with the UML. The SRF is an LRA-defined implementation of the OMG's Meta-Object Facility (MOF) [10] and uses the standard to extend the set of core model constructs. [9] The UML is also a compliant implementation of MOF so that the LRA SRF is in effect a sibling standard of UML. The SRF, like all other MOF compliant implementations, is serializable using XMI. However, XMI serialization of the SRF is intended to make parsing of LRA Technical Models easier.

The principal limitation of using the SRF is that it requires the development of specific-tooling to support it, none of which exists currently. However, it is a requirement for models represented using URF to be transformable programmatically and without loss to SRF for viewing, editing and exchange when suitable tooling becomes available. Backwards compatibility with the URF also must be maintained to ensure that dependent artefacts which are not expressed in the SRF can still be reference and have access to rendered views of LRA Technical Model artefacts in URF. This applies in particular to dependent LRA Knowledge Model artefacts which are currently expressed in UML only.

The following sections describe in turn the core UML constructs shared by the URF and the SRF and the constructs that compose the URF and SRF respectively.

3.1 Core Constructs

The following sections describe the UML class model constructs used within LRA Technical Modelling. With the exception of the realization relationship described in section 3.1.5.4, these constructs are part of the UML infrastructure library and are shared by the UML, MOF and other OMG modelling specifications. An abridged UML

class diagram of the UML elements described in the following sections is presented in appendix A.1.

3.1.1 Package

The UML package construct is used to group elements and provide a namespace for the grouped elements. [5] Within the LRA, UML packages are used to organise LRA Technical Model elements into coherent groups. Packages also serve one of two other purposes depending on the model type. For Reference Models (see section 2.1) packages provide the formal namespace (beginning with the *Ira* root package) used to construct the fully qualified name by which each model element can be unambiguously identified. For Domain Models (section 2.3) and Constrained Domain Models (section 2.4) the primary purpose of the package is to organise models into groups of elements that can be authored separately and (if required) concurrently by Technical Model authors. Domain Model and Constrained Domain Model packages are not intended to function as formal namespaces. Model Reuse Patterns (section 2.2) which serve as 'templates' for creating Domain Models and Constrained Domain Models are not organised into packages.

3.1.2 Class

The UML class is a type of UML element that has as its instances a set of objects that share the same specifications of features, constraints and semantics. [5] Within the LRA the UML class element is used to represent Domain Model and Constrained Domain Model classes (which may be instantiated) ; Reference Model classes which are not instantiated but which specify the former; and data types. A class may be declared abstract in which case it cannot have any direct instances. However any direct instance of a concrete (i.e. non-abstract) class is an indirect instance of its class's generalised class(es) even if a generalised class is abstract. Within the LRA an abstract Reference Model class implies that the class does not specify Domain Model realizations. For example, the LRA specifies the Reference Model class *Ira::technical::en13606::extract::ELEMENT* as being abstract. Consequently, Reusable Model Patterns only exist for concrete subclasses such as *PROPERTY_OBSERVATION_ELEMENT* and the class itself is not realisable directly by any Domain Model class.

A class is typically assigned a name which is used to denote the class and in the case of Reference Model classes, formally identify the class within its package. The names of Reference Model and Model Reuse Pattern classes are fixed by the Technical Model infrastructure. For Reference Model classes a qualified name can be constructed to unambiguously identify each class using the name of each package in the hierarchy of packages in which the class is nested (starting from the *Ira* root package), e.g.

Ira::technical::en13606::extract::PROPERTY_OBSERVATION_ELEMENT.

Within the LRA each class is assigned (or otherwise inherits) a natural language intentional description that describes the semantic criteria that objects must satisfy as instances of the class.

3.1.3 Property Feature

Within the UML, property instances are used to represent attributes of classes and the ends of binary associations (see section 3.1.5.3). [5] Class attributes and the navigable ends of binary associations are represented by properties that are owned by a class whereas non-navigable association ends are represented by properties owned by an association.

Within the LRA the property of each and every class or data type attribute is assigned a name specified by the LRA infrastructure. Properties representing navigable or non-navigable association ends, other than associations connected to COMPONENT_RELATIONSHIP_ELEMENTS, are assigned names specified by the LRA infrastructure.

Within the LRA each and every property has a type. The type constrains the value domain of the property and MUST represent a specified class of the same LRA model type as the class that owns the property. For example, if the property is owned by a Reference Model class then the property type MUST represent another Reference Model class.

The multiplicity of a property specifies the allowable cardinalities of an instantiation of the attribute as an inclusive interval of non-negative integers with a lower bound and an (possibly infinite) upper bound. [5] Within the LRA unless otherwise specified the multiplicity defaults to a lower bound value of 1 and an upper bound value of 1 (i.e. 1..1).

The aggregation kind of a Property specifies the type of aggregation that applies to it as an enumeration literal. The UML specifies the values for the literal as being *none*, *shared* or *composite*, with the default being *none*.

Within the LRA, a Property may specify a fixed value for the attribute.

Within the LRA each Property is assigned (or otherwise inherits) a natural language description of the attribute's function.

3.1.4 Constraint Feature

A constraint is a condition or restriction expressed in natural language text or in a machine readable language that imposes restrictions on the extension of one or more UML elements beyond those imposed by other UML language constructs. [5] The condition or restriction must be true when evaluated in order for the constraint to be satisfied.

A constraint comprises a name and a specification expressing the constraint. As well as providing a number of in-built constraints the UML allows user-defined constraints to be expressed. The syntax and interpretation of the language in which these are specified is application dependent. The LRA makes use of two constraint languages:

- Object Constraint Language (OCL) [7] is used for constraining class model structures and values; and

- an XML-based SNOMED CT Expression Constraint Language (ECL) [8] is used for constraining the semantics and / or literal form of SNOMED CT expressions specified by the instances of terminology bound classes.


A constraint is an assertion that indicates a restriction that must be satisfied by a conforming object. Although constraints expressed in natural language are important for human usage and interpretation of models, only constraints expressed using a machine readable language such as OCL or SNOMED CT ECL can be verified programmatically and / or used to validate objects.

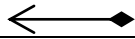
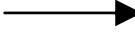
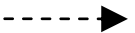
3.1.5 Relationships

Within the UML a Relationship is an abstract concept that specifies some kind of relationship between elements. [5] The specific types of UML Relationships used within LRA Technical Models are listed in Table 2.

Table 2 UML Relationships used within LRA Technical Models

Relationship type	Navigation	Model type	Source class	Target class
Association	Unspecified _____	ELEMENT Domain Model	Realization of Reference Model class Ira::technical::en13606::extract::COMPONENT_RELATIONSHIP_ELEMENT	Realization of any of: <ul style="list-style-type: none"> Reference Model class Ira::technical::en13606::extract::UNBOUND_DATA_ELEMENT concrete specialisation of Reference Model class Ira::technical::en13606::extract::BOUND_DATA_ELEMENT
	Source -> Target →	ELEMENT Constrained Domain Model	Constraint of Domain Model COMPONENT_RELATIONSHIP_ELEMENT	Constraint of Domain Model ELEMENT class
		PARTICIPATION Domain Model	Realization of Reference Model class Ira::technical::en13606::extract::FUNCTIONAL_ROLE	Constraint Type CR_Role_Choice
		PARTICIPATION Domain Model	Realization of Reference Model class Ira::technical::en13606::extract::RELATED_PARTY	Constraint Type CR_Role_Choice
		PARTICIPATION Domain Model	Realization of concrete specialisation of Reference Model class	Realization of one of:

			Ira::technical::participations::CR_Role	concrete specialisation of Reference Model class Ira::technical::participations::CR_Entity; or Reference Model class Ira::technical::participations::CR_RoleRelationship; or Reference Model class Ira::technical::participations::CR_LanguageCommunication
		PARTICIPATION Domain Model	Realization of Reference Model class Ira::technical::participations::CR_RoleRelationship	Realization of concrete specialisation of Reference Model class Ira::technical::participations::CR_Role
Association with Composition	Source -> Target 	COMPOSITION Model	Constraint of realization of Reference Model class Ira::technical::en13606::extract	Constraint of Domain Model ENTRY class
		ENTRY Domain and Constrained Domain Models	Realization of Reference Model class Ira::technical::en13606::extract::ENTRY	Realization of any of: <ul style="list-style-type: none"> Reference Model class Ira::technical::en13606::extract::UNBOUND_DATA_ELEMENT concrete specialisation of Reference Model class Ira::technical::en13606::extract::BOUND_DATA_ELEMENT
		ENTRY Domain Model	Realization of Reference Model class Ira::technical::en13606::extract::ENTRY	Constraint Type FUNCTIONAL_ROLE_CHOICE

		ENTRY Domain Model	Realization of Reference Model class Ira::technical::en13606::extract::ENTRY	Constraint Type RELATED_PARTY_CHOICE
	Target -> Source 	ELEMENT Constrained Domain Model	Constraint of Domain Model COMPONENT_RELATIONSHIP_ELEMENT class	Constraint of Domain Model ELEMENT class
Generalization	Source -> Target 	ENTRY Constrained Domain Model	Constraint of Domain Model ENTRY class	Realization of Reference Model class Ira::technical::en13606::extract::ENTRY
		ELEMENT Constrained Domain Model	Constraint of Domain Model ELEMENT class	Realization of one of: Reference Model class Ira::technical::en13606::extract::UNBOUND_DATA_ELEMENT concrete specialisation of Reference Model class Ira::technical::en13606::extract::BOUND_DATA_ELEMENT
Realization	Source -> Target 	PARTICIPATION Domain Model	Realization of Reference Model class Ira::technical::en13606::extract::FUNCTIONAL_ROLE	Constraint Type FUNCTIONAL_ROLE_CHOICE
		PARTICIPATION Domain Model	Realization of Reference Model class Ira::technical::en13606::extract::RELATED_PARTY	Constraint Type RELATED_PARTY_CHOICE

3.1.5.1 Association

A UML association specifies a semantic relationship that can occur between two or more class instances. [5] An instance of an association is called a link. Each instance in the relationship is known as an association end and is represented by a property which specifies the name, type and multiplicity of the end. In recursive relationships two or more ends have the same type.

An association may be specified as navigable or non-navigable.⁴ If an association relationship is navigable *from* a containing class then the (target) association end property is owned by the class and referenced by the association; otherwise the association is non-navigable and the association end is owned by the association itself. Non-navigable associations are underspecified within the LRA.

Within ELEMENT Domain and Constrained Domain Models the COMPONENT_RELATIONSHIP_ELEMENT MUST represent the source end of any association relationship between it and its subject or object element. This is regardless of the direction of navigability expressed by the Constrained Domain Models .

Within the LRA each association is assigned (or otherwise inherits) a natural language description that describes the function of the relationship. Named association ends optionally MAY be described.

3.1.5.2 Association with Composition

Composition is used to add the semantics of composite aggregation to an association end. The LRA uses the UML composition symbol (as shown in Table 2) to indicate an exclusive aggregation of the instance(s) represented by an association end. This is indicated by the association end property having an aggregation kind of *composite*. Any instance aggregated in this way MUST be part of one and only one composition and its lifetime MUST be coincident with the composing instance, i.e. it cannot exist before the composing instance is created or after it is deleted (unless explicitly removed beforehand). The composing instance therefore has responsibility for the existence and maintenance of its component parts. [5]

Within LRA Reference Models, the aggregation kind of *composite* is used where appropriate.

Within Domain Models and Constrained Domain Models an aggregation kind of *composite* MUST be asserted by:

⁴ Navigability, according to the UML specification [5] means instances participating in runtime links can be "accessed efficiently from instances participating in links at the other ends of the association". The actual mechanism by which access is achieved is an implementation concern. If an end is not navigable, access from the other ends may or may not be possible, and if it is, it might not be efficient. Tools operating on UML models are allowed to navigate associations from non-navigable ends.

- any property that is at the ENTRY end of an association between an COMPOSITION and an ENTRY ; and
- any property that is at the ELEMENT end of an association between an ENTRY and an ELEMENT.

Additionally within Constrained Domain Models, an aggregation kind of *composite* MUST be asserted by the property of the subject or object end of any association between a Constrained Domain Model COMPONENT_RELATIONSHIP_ELEMENT and a subject or object element.

Within the LRA, the property of each and every other association end NOT described above MUST specify aggregation kind of *none*.

3.1.5.3 Generalization Relationship

Within the LRA the UML generalization is used to assert a taxonomic 'is a' relationship between a specialised class and a more general class (i.e. generalised class) in which each instance of the specialised class is also an indirect instance of the generalised class. [5] The specialised class therefore inherits the features (including any constraints) of the generalised class and within the LRA is deemed to be usable wherever the generalised class can be used.

A generalization is always owned by the specialised class and MUST always be specified as navigable (i.e. from the specialised class to the generalised class).

3.1.5.4 Realization Relationship

Realization is a specialized type of UML dependency relationship between two sets of model elements, one representing an implementation (the source) and the other representing its specification (the target). [5] The UML does not define the meaning of 'implementation' other than to state that it implies a more refined or elaborate form with regard to a certain modelling context.

Within the LRA, the realization relationship is used to reference metadata such as the LRA Knowledge Model artefacts upon which the Technical Model artefacts depend for their specification.

The realization relationship also is used within PARTICIPATION Domain Models to indicate that a model element conforms to a choice constraint and is therefore a valid 'implementation' option for satisfying the constraint.

3.2 LRA Technical Model UML Representation

The URF uses an LRA-specific UML profile to tailor the core UML 2.1 class model constructs. A UML profile may comprise the following three constructs:

- stereotypes,
- tagged value properties, and

- profile constraints.

The following sections describe the constructs that comprise the UML Profile for LRA Technical Modelling.

3.2.1 Stereotypes

A UML profile defines a set of stereotypes, each of which extends one or more UML model element types. These model element types are known formally as metaclasses and include classes, attributes, connectors, connector ends, etc. Figure 3 illustrates how the LRA `COMPONENT_RELATIONSHIP_ELEMENT` stereotype extends the UML class metaclass.

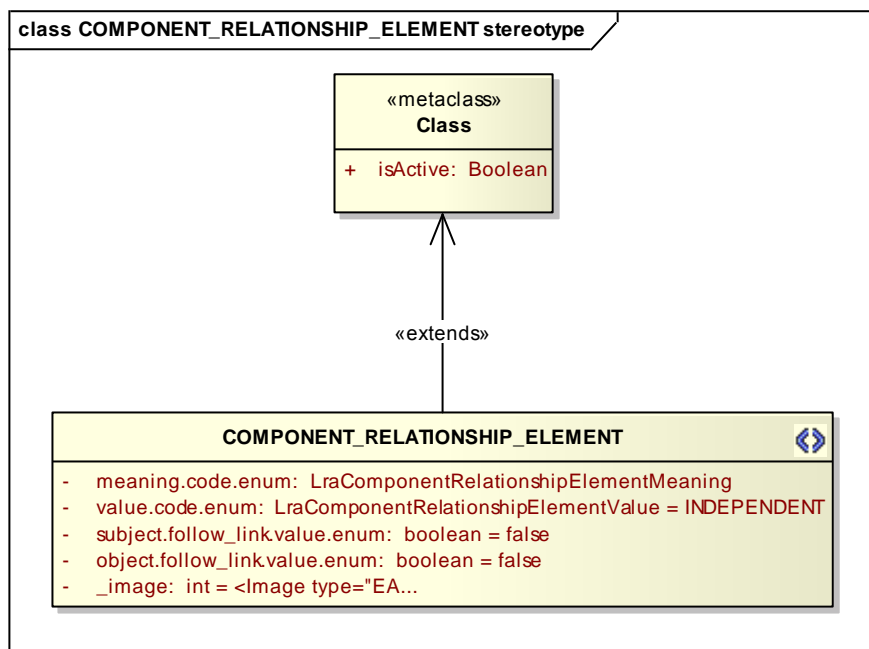


Figure 3 `COMPONENT_RELATIONSHIP_ELEMENT` stereotype

The UML Profile for LRA Technical Modelling specifies a stereotype for each Reference Model class for which Model Reuse Pattern elements and Domain Model classes can be created. Within the LRA, stereotypes are not specified for abstract Reference Model classes. Stereotypes are specified also for a number of LRA data types.

As well as elaborating the meaning of the element type(s) they extend, stereotypes are used for:

- indicating the Reference Model class to which each Domain Model or Constrained Domain Model class conforms⁵ by displaying the unqualified name of the

⁵ An alternative method of indicating the Reference Model class to which a Domain Model class conforms would be to extend a UML realization relationship (see section 3.1.5.4) from the latter to the former. Although this would use the core

Reference Model class enclosed within a pair of guillemets («») above the class name as shown in Figure 4;

- specifying the graphical representation (i.e. colour, shape and initial dimensions of model elements); and
- attaching additional, LRA-specific tagged value properties to model elements.

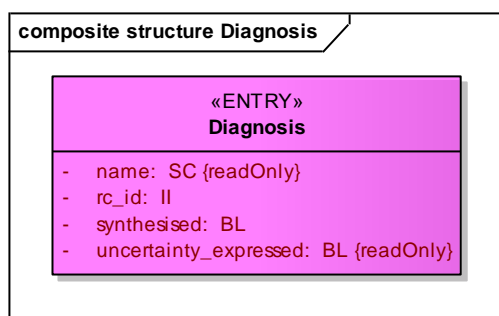



Figure 4 Example of a Technical Model class with an «ENTRY» stereotype



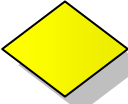
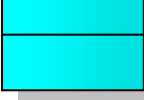

A stereotype is able to define and make available additional properties to any instance of the metaclass (e.g. a class or connector) that uses the stereotype. These additional properties are defined as attributes on the stereotype class and manifested as tagged value properties on the class that uses the stereotype. Thus every COMPONENT_RELATIONSHIP_ELEMENT class has the tags meaning.code.enum and value.code.enum amongst others associated with it. Section **Error! Reference source not found.** describes the complete set of LRA-specific tagged value properties.


Table 3 below lists the LRA defined class stereotypes grouped according to their UML renderings.

Table 3 LRA defined class stereotypes

UML notation	Rendering group	Permitted stereotype(s)
	EHR_EXTRACT and its immediate associates.	«EHR_EXTRACT» «EXTRACT_CRITERIA»

UML rather than relying on the profile extension mechanism, the downside is that it would result in considerable representational overhead.

	BS EN ISO 13606-1:2008 derived RECORD_COMPONENT classes	«COMPOSITION» «ENTRY» «CLUSTER»
	Constrained BS EN ISO 13606-1:2008 ELEMENT classes	«UNBOUND_DATA_ELEMENT» «PROPERTY_OBSERVATION_ELEMENT» «FINDING_OBSERVATION_ELEMENT» «INVESTIGATION_ACTIVITY_ELEMENT» «SUBSTANCE_ACTIVITY_ELEMENT» «GENERAL_ACTIVITY_ELEMENT» «RECORD_ARTEFACT_ELEMENT»
	COMPONENT_RELATIONSHIP_ELEMENT with associated LINK instances	«COMPONENT_RELATIONSHIP_ELEMENT»
	Extract participations and other classes	«FUNCTIONAL_ROLE» «RELATED_PARTY» «CR_Participation» «AUDIT_INFO» «ATTESTATION_INFO»
	Role classes	«CR_RolePerson» «CR_RoleOrganisation» «CR_RoleIncidentalDevice» «CR_RoleIncidentalLocation» «CR_RoleRelationship»

	Entity classes	«CR_EntityPerson» «CR_EntityOrganisation» «CR_EntityDevice» «CR_EntityPlace»
---	----------------	---

The graphical notations for the «COMPONENT_RELATIONSHIP_ELEMENT» stereotype shown in Table 3 are in fact abbreviations of a compositional structure consisting of a COMPONENT_RELATIONSHIP_ELEMENT class and its two associated LINK instances as shown in Figure 5. For representational convenience the attributes of both classes are represented using tag value properties.

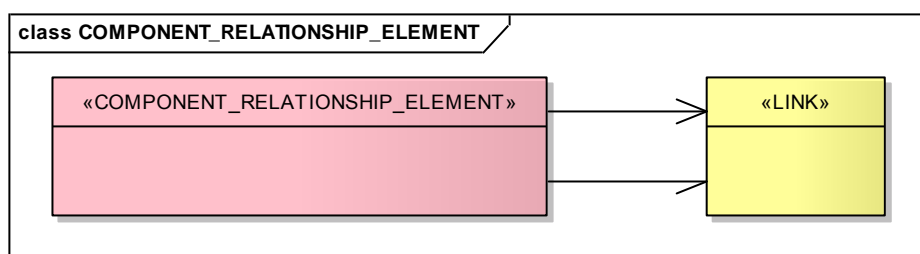
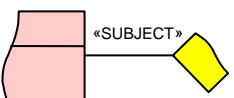
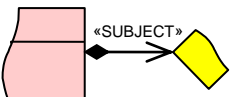
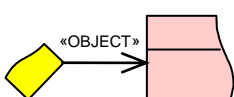


Figure 5 Expanded COMPONENT_RELATIONSHIP_ELEMENT class model

Table 4 below lists the LRA defined relationship stereotypes grouped according to their UML renderings.

Table 4 LRA defined relationship stereotypes

UML notation	Relationship	Permitted stereotype(s)
	Domain Model component relationship association	«SUBJECT» «OBJECT»
	Constrained Domain Model component relationship subject association	«SUBJECT»
	Constrained Domain Model component relationship object association	«OBJECT»

3.2.2 Tagged Value Properties

The attributes defined by a stereotype, as illustrated in Figure 3, are made available as tagged value properties to model elements that use the stereotype. A tagged value property comprises a named tag and a value. Thus classes with stereotype «COMPONENT_RELATIONSHIP_ELEMENT» have meaning.code.enum,

value.code.enum and other tagged value properties which derive from the attributes of the same name defined by the stereotype. The complete set of LRA specific tags are described in section **Error! Reference source not found.**

3.2.3 Profile Constraints

A profile constraint specifies rules about how and where stereotyped model elements and their tagged values can be used in a model. Profile constraints may be used for instance to specify the permitted associations between different stereotyped element types. Currently the UML Profile for LRA Technical Modelling does not specify any profile constraints.

3.3 LRA Technical Model Standard Representation

The URF uses the UML profile mechanism. This provides a lightweight mechanism for extending the UML with constructs specific to a particular domain, platform, or method. Profiles enable new constraints to be added to the UML metamodel but they do not permit the addition of new metaclasses or relationships to the metamodel or the removal of UML specified constraints.

The SRF, however, uses MOF [10] which is designed to facilitate the creation of user defined metamodels that specify their own metaclasses, relationships and constraints. The UML also is a MOF defined metamodel.

4 LRA Technical Model Constraint

This section describes the various types of Technical Model constraint and their methods of specification. The term ‘constraint’ is used here in its broadest sense to mean any model construct that asserts a restriction upon the semantics of one or more other model constructs. This definition includes and goes beyond the use of the UML constraint feature described section 3.1.4.

The constraint types described are intended to be independent of representational form. The way in which some types of constraint are specified, however, does depend on the particular representational form. Where this is the case the details of the method of specification for the particular representational form are described in an additional sub-section.

4.1 Class and Data Type Level Constraints

This section specifies types of constraint that may be applied to class constructs and their methods of specification, which in some cases depends on the representational form.

4.1.1 Restriction of Intension

The natural language intentional description of a class, data type or data type flavor MAY be restricted by the intension asserted by any realising class, or specialising class or data type, or by any further flavor constraint (as applicable). This may be achieved, for example, by extending the semantic criteria that objects must satisfy as instances of the realising class or specialising class or data type.

Restriction of intension may be applied by:

- Reference Model specialisations of other Reference Model classes;
- data type specialisations of other data types;
- data type flavors of data types or other data type flavors;
- Domain Model realisations of Reference Model classes; and
- Constrained Domain Model specialisations of Domain Model classes.

Restriction of intension of Reference Model classes and data types by their respective Reference Model class and data type specialisations is predefined by the LRA Technical Model infrastructure.

For Domain Models and Constrained Domain Models, the intension asserted by any realising or specialising class or by any data type flavor MUST NOT extend or conflict with the intension asserted by the more general class, data type or data type flavor to which the constraint is being applied.

4.1.2 Class and Data Type Specialisation

Within the LRA, specialisation involves the creation of a specialised class (or data type) from a more general class (or data type). This is done by asserting a generalization relationship (see section 3.1.5.3) that extends from the specialised class (or data type) to the generalised class (or data type). This implies that the specialised class inherits all of the features (including attributes, relationships and constraints) of the generalised class and can substitute for it.

Specialisation is applicable to:

- specialisation of a Reference Model class by another Reference Model class;
- specialisation of a data type by another data type; and
- specialisation of a Domain Model class by a Constrained Domain Model class.

Reference Model class and data type specialisations are predefined by the LRA Technical Model infrastructure and MAY restrict the semantics inherited from the classes or data types they specialise, for example by redefining properties (see section 4.2) or asserting additional constraints (see section 4.1.5). Specialisation is also the mechanism used to extend the semantics of specialised Reference Models classes or data types by adding new properties. Reference Model and data type specialisation relationships MAY include multiple inheritance.

A Constrained Domain Model class MAY restrict the inherited semantics of the Domain Model class it specialises. For example it may restrict the intension or redefine properties of the generalised class (see sections 4.1.1 and 4.2) or assert additional constraints (see section 4.1.5). However, a Constrained Domain Model class MUST NOT extend the semantics of the Domain Model class it specialises, for example by extending its intension, specifying additional properties or relaxing constraints. Multiple inheritance is not permitted for Constrained Domain Model classes. Therefore a Constrained Domain Model class MUST be the source of one and only one generalization relationship.

4.1.3 Reference Model Class Realization

Within the LRA, Reference Model class realization involves the creation of a Domain Model class conforming to a specified concrete Reference Model class. This process is facilitated using Model Reuse Patterns, as described in section 2, which act as templates to ensure that all new Domain Model classes conform to the Reference Model.

With the exception of COMPONENT_RELATIONSHIP_ELEMENTS, which are anonymous, a newly created Domain Model class MUST be assigned a name to denote informally its purpose or nature. The name might also serve as a mnemonic for the class.

A Domain Model class MUST NOT extend the semantics of the Reference Model class it realises by for example extending its intension, specifying additional

properties or relaxing constraints. A Domain Model class **MUST** explicitly restate all of the attributes of the Reference Model class it realizes. Because all of its attributes are restated (rather than inherited), a Domain Model class **MAY** explicitly constrain any non-fixed attribute as described in section 4.2

4.1.3.1 URF for Reference Model Class Realization

The relationship between a Domain Model class and the Reference Model class it realises is indicated by stereotyping the former with the unqualified name of the latter. In realising a Reference Model class a Domain Model class **MUST** restate all attributes and **MAY** apply constraints to one or more.

4.1.4 Data Type Flavors

The LRA uses data type flavors for defining named constraints on existing on ISO/FDIS 21090 data types as specified by the HL7 v3 *Data Types - Abstract Specification, Release 2* [6]. A data type flavor does not add any new semantics (including properties or setting of default values) to a data type, but constrains it for a particular purpose. Furthermore a flavor **MAY** constrain one or more other flavors in which case the constraints specified by the new flavor are added to those already specified in the other flavor(s).

4.1.5 Ad hoc Constraints

Any realising class, or specialising class or data type, or any flavor **MAY** assert additional model constraints. These constraints are expressed using OCL and are an addition to the standard property level and value space constraints described in the sections below. They are ad hoc in the sense that they are typically asserted to specify a particular configuration of instances within a model, e.g. specifying the co-occurrence of association end instances and / or attributes.

Ad hoc constraints may be applied by:

- Reference Model specialisations of other Reference Model classes;
- data type specialisations of other data types;
- data type flavors of data types or other data type flavors;
- Domain Model realisations of Reference Model classes; and
- Constrained Domain Model specialisations of Domain Model classes

Ad hoc constraints asserted by Reference Model classes and data types and their specialisations are predefined by the LRA Technical Model infrastructure.

For Domain Models and Constrained Domain Models, any additional constraint asserted by realising or specialising class or by any data type flavor **MUST NOT** extend or conflict with any constraint asserted by the more general class, data type or data type flavor to which the constraint is being applied.

4.2 Property Level Constraints

This section specifies types of constraint that may be applied to the properties of classes and data types, and their methods of specification. The method of specification depends upon the Technical Model type and in some cases on the representational formalism.

Domain Model realizations explicitly restate the attributes of the Reference Model class to which they conform. As a result a Domain Model class is able to constrain directly the function, type or occurrence or specify the fixed value of any non-fixed attribute.

For all other model types, redefinition as defined by the UML [5] is used to assert constraints on the inherited properties of specialized classes and data types, and data type flavors as the inherited properties cannot themselves be changed. Redefinition MAY therefore be used by any of the following to assert property level constraints:

- Reference Model specialisations of other Reference Model classes;
- data type specialisations of other data types;
- data type flavors of data types or other data type flavors;
- Domain Model realisations of Reference Model classes; and
- Constrained Domain Model specialisations of Domain Model classes.

Restatement and redefinition both enable constraints to be applied one or more parts of the property of a class or data type attribute as described in the following sub-sections. Property redefinition also allows constraints to be applied to the end of a binary association between two Constrained Domain Model classes.

Redefinition of the properties of Reference Model class and data type attributes by their respective Reference Model class and data type specialisations is predefined by the LRA Technical Model infrastructure.

4.2.1 Restriction of Attribute Function

The function, as described in natural language, of an attribute of a class or data type MAY be restricted by the function asserted for the attribute by any realising class or specialising class or data type (as applicable).

The function asserted for an attribute by any class, data type or data type flavor MUST NOT extend or conflict with the function asserted for the attribute by the class, data type or data type flavor to which the constraint is being applied.

4.2.2 Constraining Property Type

The property type constrains the value domain of an attribute or association end. For Reference Model classes, the property type of an attribute MAY be restated and

constrained by a Domain Model class; for classes, data types or data type flavors within a specialization hierarchy the property type of an attribute MAY be redefined by any specialising class, data type or constraining flavor (as applicable). A constrained property's type MUST be a realization or specialisation of the property type specified by the more general class, data type or data type to which the constraint is being applied. For example the Reference Model class `PROPERTY_OBSERVATION_ELEMENT` specifies a value attribute of property type `ANY`. A Domain Model realization of the class may redefine this to any property type that has the `ANY` data type as its generalised class which in this case means that it can be substituted with any LRA ISO/FDIS 21090 conformant data type (e.g. `PQ`, `INT`, `BL`, etc).

For Constrained Domain Models, the property type of the end of an association between two classes always redefines the property type of the corresponding end of the more general association between the two Domain Model classes being specialised.

4.2.2.1 URF for Attribute Property Redefinition

The URF uses a standardised OCL expression type, as shown below, to specify redefinition of the property type of an inherited attribute.

Constraint type: LRA Invariant (Type Redefinition)

Constraint name:

Syntax	'attribute' <attribute name> 'is of type' <redefining data type or primitive type>
--------	--

Example	attribute value is of type <code>IVL.HIGH<PQ></code>
---------	--

	attribute code is of type <code>SnomedCtExpression</code>
--	---

Constraint specification:

Syntax	'inv:' <attribute name>'.oclIsTypeOf('<redefining data type or primitive type>')
--------	--

Example	inv: value.oclIsTypeOf(<code>IVL.HIGH<PQ></code>)
---------	---

	inv: code.oclIsTypeOf(<code>SnomedCtExpression</code>)
--	--

4.2.3 Constraining Occurrence

The lower bound of the multiplicity (m_0) of an attribute of a Reference Model or Domain Model class MAY be restricted by the lower bound of the multiplicity (m_1) specified by any realising or specialising class such that $m_1 \geq m_0$ where m indicates

a non-negative integer. Changing the lower bound multiplicity of an attribute from zero to a value greater than zero indicates that the attribute is defined and in the instance MUST specify a minimum number of values equal in number to the lower bound.

The upper bound of the multiplicity (n_0) of an attribute of a Reference Model or Domain Model class MAY be restricted by the upper bound of the multiplicity (n_1) specified by any realising or specialising class such that $n_1 \leq n_0$ where n indicates an unlimited natural number greater than or equal to m . Changing to 0 the upper bound multiplicity of an attribute whose lower bound multiplicity is 0 (i.e. resulting in 0..0) indicates that the attribute is undefined and that its use is therefore prohibited.

4.2.3.1 URF for Occurrence Constraints

Domain Model realizations MAY explicitly redefine the multiplicity of one or more Reference Model attributes. This includes setting the upper bound to zero (provided the lower bound multiplicity is zero) and thereby prohibiting use of the attribute.

For specifying Constrained Domain Model occurrence constraints the URF defines two standardised OCL expression types. For any occurrence constraint in which the attribute is defined (i.e. upper bound multiplicity > 0), the expression takes the following form.

Constraint type: LRA Invariant (Occurrence Constraint)

Constraint name:

Syntax 'attribute' <attribute name> 'has minimum occurrence' <m> 'and maximum occurrence' <n>

Example attribute obs_time has minimum occurrence 1 and maximum occurrence 1

Constraint specification:

Syntax 'inv:' <attribute name>'->size() >=' <m> 'and' <attribute name>'->size() <=' <n>

Example -- the constraint specifies explicitly both the lower and upper bound multiplicities

inv: obs_time->size() >= 1 and obs_time->size() <= 1

OCL allows any attribute or association end with an upper bound multiplicity of one to be treated as a Set containing a single instance. This enables OCL collection or set properties such as size() to be accessed using the arrow notation as shown in the constraint description above. [7]

For any occurrence constraint in which the attribute is undefined (i.e. upper bound multiplicity = 0), the OCL operation `oclIsUndefined` is used as shown below.

Constraint type: LRA Invariant (Use Of Attribute Prohibited)

Constraint name:

Syntax 'use of attribute' <attribute name> 'is prohibited'

Example use of attribute `obs_time` is prohibited

Constraint specification:

Syntax 'inv:' <attribute name>'.oclIsUndefined()'

Example inv: `obs_time.oclIsUndefined()`

4.2.4 Asserting Fixed Values

An attribute of a Reference Model class, Domain Model class or data type, whose value is not already fixed, MAY be set to a status of read only by any realising class, specialising class or data type flavor (as applicable). Within the LRA, setting a Reference Model or Domain Model class attribute value to fixed involves assigning a value to the attribute and then setting the UML `isReadOnly` status to true. The value of any attribute with a status of read only is fixed and MUST NOT be changed by any realising class, specialising class, constraining flavor or runtime instance.

Flavors of ISO/FDIS 21090 data types MAY specify fixed values. [6].

4.2.4.1 URF for Fixed Value Assignment

In addition to setting the `isReadOnly` status to true, the URF requires that a single value be assigned to the enum tag (see section 4.3) of each primitive type attribute whose value is to be fixed within the data type instance.

4.2.5 Constraining Navigability

Navigability, in one direction only, MUST be asserted for the property of the end of an association between a Constrained Domain Model

`COMPONENT_RELATIONSHIP_ELEMENT` and a subject or object element.

Subject associations MUST be navigable from the subject element to the `COMPONENT_RELATIONSHIP_ELEMENT`; object associations MUST be navigable from the `COMPONENT_RELATIONSHIP_ELEMENT` to the object element.

Regardless of the direction of navigation, the

`COMPONENT_RELATIONSHIP_ELEMENT` always represents the source end of the association relationship. Navigable subject association relationships are therefore represented as navigable from target to source.

If navigability is asserted from the COMPONENT_RELATIONSHIP_ELEMENT to the subject or object element then the property representing the navigable end of the association is owned by the COMPONENT_RELATIONSHIP_ELEMENT. If navigability is asserted in the opposite direction then the subject or object element becomes the owner of the property representing the navigable end. In both cases, and for

4.2.5.1 URF for Component Relationship Navigability

The URF for component relationship is shown in within Domain Models and Constrained Domain Models is shown in Table 4 of section 3.2.1.

4.3 Value Space Constraints

A value space constraint is a machine readable expression that imposes restrictions on the extension of one or more UML primitive types (specified under the `lra::datatype::uml` package). Value space constraints are used to constrain the value domain of the primitive type instances that compose the ISO/FDIS 21090 derived data types used by LRA Technical Model attributes

The LRA defines a number of distinct types of value space constraint (see Table 5) each of which defines a particular type of restriction and applies to one or more specific primitive types and their specialisations.

Table 5 Value Space Constraint Definitions

Value space constraint type	Description	Constraint Primitive Type	Rendered constraint name	Primitive Type applied to							
				Boolean	Code	Integer	Real	SnomedCtExpression	String	Timestamp	UnitOfMeasure
codeSystemRef	A reference to the code system to which a specialised instance is bound and to which any set of enumerations MUST therefore conform as a subset.	Uid	Code system (referred to by the Uid)		■						
default	A default value.	Primitive ⁶	Default value	■	■	■	■	■	■	■	■
enum	A set enumeration literals that specifies the set of valid values of the instance. An enumeration set with one member indicates a fixed value.	Primitive	Literal value(s)	■	■	■	■	■	■	■	■
expression	A reference to a constraint expression which restricts the possible ways that an instance of a SNOMED CT terminology expression may be constructed to represent a given meaning.	Uuid	Literal expression constraint (referred to by					■			

⁶ As UML primitive type to which the constraint applies.

Page 40 of 48

	binary type.											
regex	A regular expression pattern that the literal value of an instance must match.	String	Regular expression	■	■	■	■	■	■	■	■	■
semantic	A reference to a constraint expression which restricts the possible meanings that can be expressed by an instance of a SNOMED CT expression.	Uuid	Semantic expression constraint (referred to by the Uuid)					■				
totalDigits	The maximum number of digits in an instance of a numeric type.	Non-negative Integer	Total number of digits			■	■					

4.3.1 URF for Value Space Constraints

The URF uses tagged values to specify value space constraints. These tags are applied to LRA primitive types, data types and Technical Model classes. When applied to a primitive type, the tag is named after the tag type name (as Table 3). When applied to a data type, the tag name takes the name of the primitive type attribute of the data type and the tag type name. When applied to a Technical Model class, the tag takes the name of the model attribute, the primitive type attribute of the model attribute's data type and the tag type name.

The following list shows the names of the value space tags defined for primitive types:

code.codeSystemRef	extension.enum	value.enum
code.default	extension.regex	value.fractionDigits
code.enum	nullFlavor.default	value.maxExclusive
code.expression	nullFlavor.enum	value.maxInclusive
code.semantic	root.default	value.maxLength
codeSystem.default	root.enum	value.length
codeSystem.enum	root.regex	value.minExclusive
codeSystemName.enum	unit.default	value.minInclusive
codeSystemVersion.default	unit.enum	value.minLength
codeSystemVersion.enum	unit.quantityKind	value.regex
extension.default	value.default	value.totalDigits

4.4 LRA Technical Model Derivation

Table 6 lists the constraint types used (or available to use) to derive successive instances of each Technical Model type as described in section 2 (excluding Model Reuse Patterns).

Table 6 Constraint types used (or available to use) to derive Technical Model type instances

Constraint type	Section	Model derivation		
		LRA Reference Models from imported reference models	Domain Models from LRA Reference Models	Constrained Domain Models from Domain Models
Restriction of intension	4.1.1	✓	✓	✓
Class and data type specialisation	4.1.2	✓		✓
Reference model class realization	4.1.3		✓	
Specification of data type flavors	4.1.4	✓		
Specification of ad hoc constraints	4.1.5	✓	✓	✓
Restriction of attribute function	4.2.1	✓	✓	✓
Constraint of property type	4.2.2	✓	✓	✓
Constraint of occurrence	4.2.3	✓	✓	✓
Assertion of fixed value	4.2.4	✓	✓	✓
Value space constraint	4.3		✓	✓

5 LRA Technical Model Serialization

This section specifies the SRF Metamodel and schema for serializing instances of LRA Technical Model types.

5.1 Data, Models and Metamodels

Data is some representation of actual data values such as a blood pressure reading of 120 over 80 that might populate a database field, data entry sheet or runtime object. A model is a formal specification of the content, structure, function and / or behaviour of conforming data instances. A set of data values representing a blood pressure data instance might for example be represented using an LRA ENTRY Constrained Domain Model or an NPfIT HL7 v3 CDA template or an openEHR archetype. These are all models designed to represent any instance of blood pressure data.

A metamodel is in turn a model of a model: that is, a formal specification of the content, structure, function and / or behaviour of conforming model instances. Thus the UML, openEHR specifications and the LRA SRF all define metamodels which enable the construction of conformant models. In the case of the UML and LRA SRF, however, the metamodel specifications are derived from the MOF which may therefore be regarded as a meta-metamodel.

Data instances, models and metamodels thus comprise a meta-level information hierarchy in which each level is a formal specification for conforming instances (be they data or models) of the level below. The example above describes three meta-levels above the level of instance data (see Figure 6). However, a meta-level hierarchy may define any number of levels and MOF, which is defined using UML can be used to specify many different types of data or model representations at any level in the hierarchy.

Meta-level	Example(s)
Meta-metamodel	MOF
Metamodel	UML, LRA SRF
Model	ENTRY Constained Domain Model: BloodPressure
Data	Blood pressure reading of 120/80

Figure 6 Example Meta-level Information Hierarchy

5.2 SRF Metamodel Requirements and Design Rationale

The MOF-based SRF Metamodel is designed to address the following key requirements.

1. The SRF Metamodel SHALL support in full all LRA Technical Model types⁷ and their semantics (including for example but not limited to redefinition of class attributes and value space constraints for static models).
2. Serializations of Technical Model artefacts conforming to the SRF Metamodel SHALL be transformable to and from URF and is doing so shall maintain traceability to the LRA Knowledge Model input requirements to LRA Technical Models.
3. The SRF Metamodel SHALL provide for the integration of multiple standard serialization technologies so that different parts of the Technical Model artefact content can be serialized using the most appropriate technology. This to avoid inventing custom serializations for say structural constraints or documentation, and instead takes advantage of existing technologies that satisfy comprehensively the serialization requirement, have international recognition and may already be in widespread use with accompanying tooling support.
4. Serializations of Technical Model artefacts conforming to the SRF Metamodel SHALL be amenable to automated transformation.
5. The SRF Metamodel SHALL facilitate specification of computable structural and terminology constraints in serialized conformant Technical Model artefacts.
6. The SRF Metamodel SHALL facilitate the identification, versioning and change control of Technical Model artefacts.

XMI (XML Metadata Interchange) [12] is the OMG standard used to facilitate the interchange of MOF conformant models via XML documents. [13] Further details concerning the rationale for using XMI as the primary serialization technology for LRA Technical Model artefacts is presented elsewhere. [14]

The classes, relationships and constraints that comprise the SRF Metamodel are defined under Reference Model package `Ira::artefacts`. The artefacts package contains four immediate sub-packages that define model constructs representing:

- formal, computable definitions or exemplars such as models, constraints, schemas and example instances;
- metadata including natural language descriptions;
- referential artefacts that enable the unambiguous referencing of artefact instances; and
- artefacts that facilitate the management and recording of change.

⁷ The metamodel specification for LRA Technical Model query definitions is not elaborated in this version of the document as specification of the Technical Model query infrastructure is still ongoing at the time of writing.

The components of the SRF Metamodel are organised in such a way so as to allow the construction of BS EN ISO 13606-1:2008 conformant models in general and models that conform to the BS EN ISO 13606-1:2008 derived (with extensions) LRA Technical Model infrastructure in particular.

TO BE COMPLETED

6 Appendices

A.1 LRA Core Modelling Constructs

Figure 7 shows an abridged class diagram of the UML InfrastructureLibrary classes, which define the core UML modelling constructs used by both the UML Representational Form (URF) and the Standard Representational Form (SRF) of the LRA.

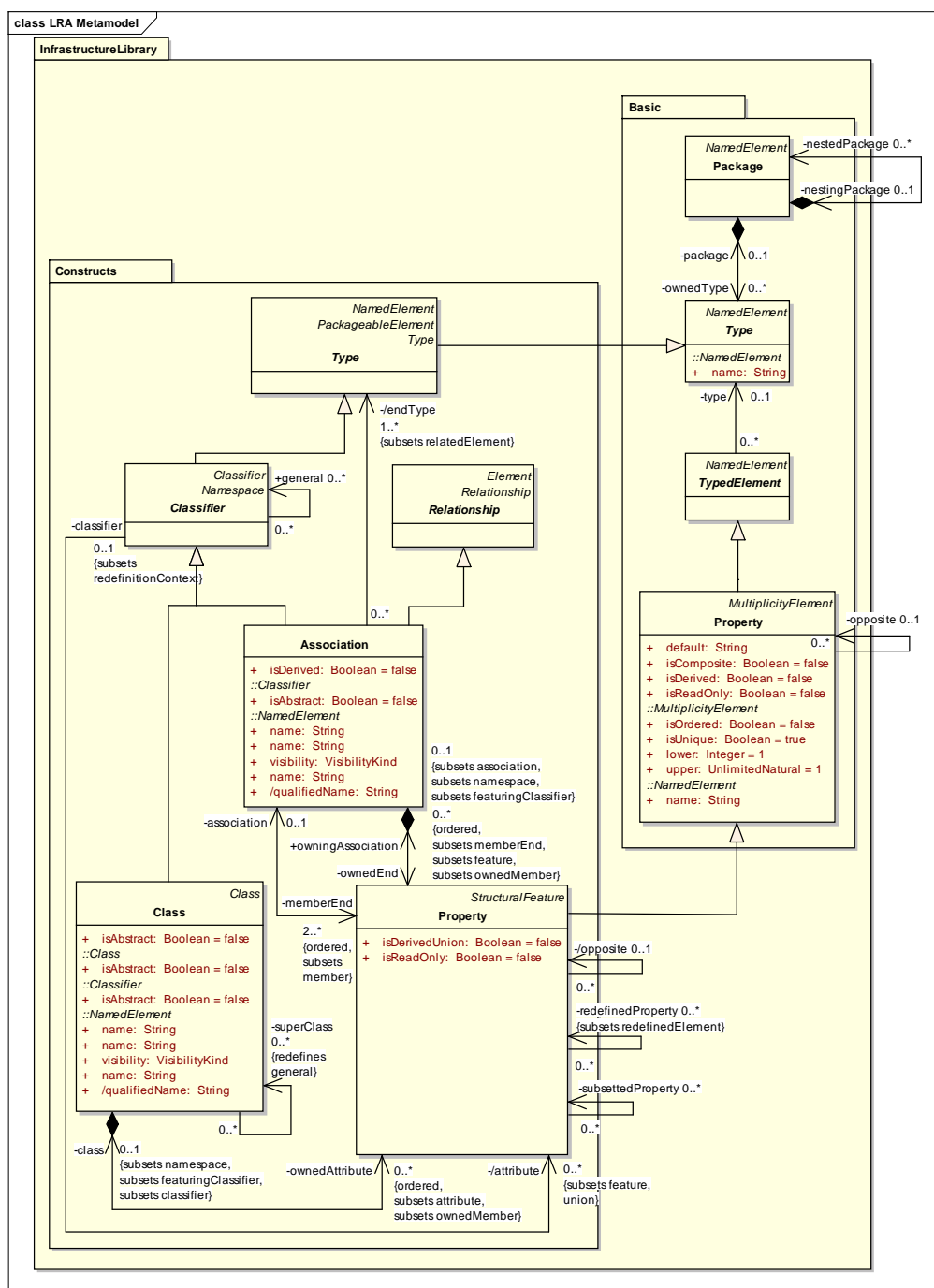


Figure 7 Abridged class diagram of the UML InfrastructureLibrary classes

