The LCC Entity Model

Version 1.0, April 2013
Editor Godfrey Rust, Metadata Workstream Leader godfrey.rust@rightscom.com

Contents

1 Introduction

1.1 Overview

The LCC Entity Model is a deliverable of the Linked Content Coalition ("LCC") project for interoperability of intellectual property rights data. This is version 1.0 of the Model, published as part of the LCC Framework in February 2013.

Section 2 provides the formal specification of the model.

Section 3 lists the mandatory **Allowed Value Sets** (controlled vocabularies) used in the model.

Section 4 summarises some ways in which the model can be used and optimized.

A UML diagram for the LCC Entity Model is included, and available as a .JPG document (*The LCC Entity Model*) and in an .EAP file (*The LCC Rights Reference Model)* provided separately.

Supporting this document is a draft of a formal representation of the LCC Entity Model as an XML schema, *LCCCommonRightsFormat* (XSD).

The Entity Model is used as the basis of the LCC Rights Reference Model (RRM) specified in the document *The LCC Rights Reference Model*.

1.2 Background

The LCC Entity Model is a direct descendant of the <indecs> Metadata Framework[1], and may be viewed as a formalization of a part of that Framework tested and refined over more than a decade in commercial and standards work. It has informed and been informed by a number of content sector metadata standards including ONIX, DDEX and the DOI.

1.3 Conventions

Naming and presentation conventions used in this document are as follows:

- Words with capital letters (eg Category, Link) are LCC-defined terms, normally expressed with the prefix *lcc:*.
- Term names are presented as a single string without spaces (eg DescriptorPart, not Descriptor Part) to make clear that a single term is meant.
- Names of AllowedValueSets (that is, LCC controlled vocabularies) are constructed with "_AVS" at the end (eg *TimeMode_AVS*).
- Names of LinkTypes and Relators are constructed by combining the names of the two roles being joined, separated by an underscore (eg *lcc:Creation_Creation, lcc:Right_Party*).
- Namespaces to which terms belong are indicated by prefixes (eg *lcc*:). The prefix *"xyz*:" is used as a "dummy" prefix for terms used in examples.

[1] *The <indecs> metadata framework Version 2.0*, June 2000

2 Specification

2.1 **Overview**

The **LCC Entity Model** is a generic, modular metadata model which provides the building blocks from which full data models such as the **LCC Rights Reference Model** (RRM) can be built. The RRM is therefore a "Profile" of the LCC Entity Model, in which additional domain-specific constraints are layered onto the Entity Model to create a set of Entity Types suitable for representing intellectual property rights.

2.1.1 **Principles of the Entity Model**

The Entity Model is a template from which data models for any types of Entity may be modelled. It classifies every Attribute of an Entity as one of five kinds (Category, Descriptor, Quantity, Time or Link). Each of these has its own "micro-model" or template, formally described in this document.

Conventionally, the only "type" constraints which apply to attributes of an Entity are those of formal datatypes (for example, string, integer, Boolean, URI, XML enumeration etc), which vary according to the physical schema employed. These restrict the formal structure of the Attribute's value and may have complex forms (such as alphanumeric sequences or checksums) but rarely have any inherent semantics[2].

The LCC Entity Model does not replace these, of course, but provides in addition a layer of richer "datatypes" which explicitly recognise the following meanings of element values in a consistent form across all Entities:

- names
- identifiers (names unique in their domain)
- annotations
- types of any Attribute value
- parts of any Attribute value
- designations (different representations of the same element value) of any Attribute value
- measurements: single numeric values and ranges, units of measure and reference units, accuracy
- times: time points and periods, accuracy
- links, and roles played by Entities in links, and the sequencing of links

Further, recognising each Attribute as an Entity in its own right (see 2.1.3) allows these meanings to be proliferated to any level of granularity or complexity.

2.1.2 **Benefits and disadvantages of the Entity Model**

There are several obvious benefits arising from the use of the Entity Model:

- Applications may be built using a relatively small number of modules which can be applied to data covering all types and combinations of Entities.
- Systems can be "data driven" and extended by parameterization, without requiring changes to underlying schemas.

[2] It may be suggested that datatypes have no inherent semantics, but there is obvious inherent meaning in the values of some common datatypes such as Boolean, and in some more complex datatypes (for example, specifying the sequence of characters and punctuation in an identifier such as ISBN or ISRC) certain elements have defined meanings.

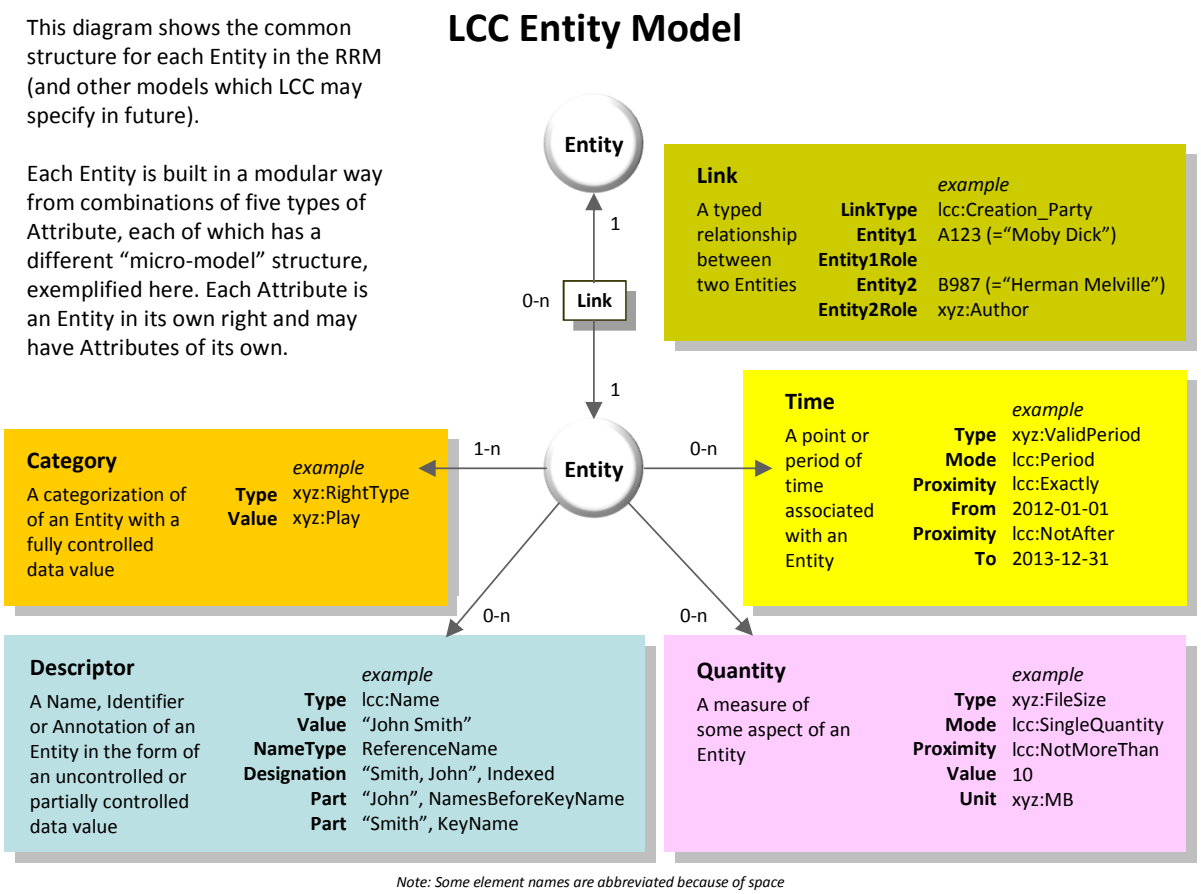
- Data in more restricted (or "optimized") models can be transformed into the richer form of the Entity Model for aggregation with data from other sources, or for transformation into other more restricted forms[3].

For systems analysis, the Entity Model also provides a comprehensive "checklist" which can be used to test the completeness of any given model.

The downside of the Entity Model is simply that in its "full" form it contains more levels of indirection than more "flattened" or "optimized" models, and therefore systems built on the full model will typically be less efficient[4]. This weakness is offset by rules which allow for the optimization of the model in specific applications by "flattening" and/or "displacing" Attributes.

2.1.3 Attribute Types

To meet the requirements of flexibility and extensibility, the LCC Entity Model takes a modular approach to modelling Entities. Any Entity may have any number of Attributes of each of five types, defined in Table 1 and shown graphically in Figure 1.



Note: Some element names are abbreviated because of space

Figure 1: LCC Entity Model overview

Table 1: LCC Attribute Types

[3] This is the main focus of the RDI (Rights Data Integration) project which is implementing the LCC Rights Reference Model.
[4] To a greater or less extent, of course, this weakness can also be offset by processing power and data capacity, and more flexible, generic models like the LCC Entity Model are now increasingly common in large systems.

Attribute Type	*Definition*	*Examples*
Category (see 2.2)	A categorization of an Entity with a fully-controlled data value.	*RightType = xyz:Copy* *Language = iso3166-1a2:EN ("English")* *WorkType = xyz:AudiovisualWork*
Descriptor (see 2.3)	A name, identifier or annotation of an Entity in the form of an uncontrolled or partially controlled data value.	*Name = "Tom W Brown"* *ISBN = "9740946014632"* *Note = "Do not open until Christmas"*
Quantity (see 2.4)	A measure of some aspect of an Entity.	*PercentageShare = 50%* *NumberOfUses = 25* *Height = 22 cm*
Time (see 2.5)	A point or period of time associated with an Entity.	*PeriodStartTime = 2013-01-03* *DateOfCreation = 1975*
Link (see 2.6)	A typed relationship between two Entities.	"Moby Dick" has the Author "Herman Melville", modelled as a Link between two Entities representing "Moby Dick" and "Herman Melville" with *LinkType = lcc:Creation_Party* and *Entity2Role = xyz:Author.*

Values of each of the first four Attributes may each be represented in multiple forms or **ValueDesignations** (see 2.3.2). Descriptor values may be subdivided into **ValueParts** (see 2.3.3), which can be assembled by rules into different ValueDesignations.[5]

Each of these Attributes has its own "micro-model", and the Types of individual Attributes can be specialized through the use of Categories with values drawn from Allowed Value Sets.

An Attribute may be an Entity in its own right, and therefore have its own Attributes. By this means the model is extensible to any level of detail or complexity. For example, a Descriptor might have a Category of *Language,* a Period during which it is valid, a Link to its author, a Quantity showing the number of words it contains and a further Descriptor in the form of a note or comment about it. Each of these Attributes may in turn have Attributes of their own.

2.1.4 **Logical model**

The logical model of an Entity is described in Table 2.

Table 2: Logical elements of an Entity

Element	*Description*	*occ*	*Example, comment*

[5] All Attribute values may be broken into Parts, in principle to any level of granularity. Each Part belongs to the same Attribute Type as its parent (for example, the "Month" part of a Time is a Time in its own right) and so the whole and part can be associated with a Link (with a suitable *xyz:Whole_Part* LinkType), so this does not require any further extension of the Entity Model. An exception is made for DescriptorPart, which is introduced in form of an element in the logical model (see Table 4) as an "optimization" of a Link into a Descriptor, simply because of its widespread use.

Category (see 2.2)	A Category of the Entity. An Entity must have one EntityType.	1-n	eg *Language = iso3166-1a2:EN ("English")*
Descriptor (see 2.3)	A Descriptor of the Entity.	0-n	eg *Title = "Moby Dick"*
Quantity (see 2.4)	A Quantity of the Entity.	0-n	eg *Number of Pages = "Approximately 320"*
Time (see 2.5)	A Time of the Entity.	0-n	eg *Date Of First Publication = "1851".*
Link (see 2.6)	A Link between the Entity and another Entity.	0-n	eg X *Creator_Party* Y

In a physical model, an Entity will typically be denoted by an ID, such as a primary or foreign database key (in a relational system) or a URI (in linked data), or by some other means according to the formalism of the system, such as the element nesting in an XML schema.

The types of Attribute which are valid for each EntityType are determined by constraint rules: for example, an Entity with *EntityType =lcc:Party* has exactly one Category of *PartyType.* These rules may be of any complexity and expressed in any logical language as required. The Entity Model has a small set of constraint rules (typically expressed as cardinalities on specific elements in the logical models of each Attribute) which apply to all profiles: more specialized rules may then be added to any extent in other profiles such as the RRM, so long as they don't conflict with the core rules of the Entity Model.

2.1.5 Optional, mandatory and excluded Attributes ("OR, XOR, AND, NOT")

By default, any Attribute of an Entity is assumed to be valid and included in the scope of the Entity. For example, if *Language = iso3166-1a2:EN* and *Title = "Moby Dick"*, then the Creation is in English (but not in any other language) AND it is called "Moby Dick" (but not anything else). This principle also applies to Attributes of the same type, so if *Language = iso3166-1a2:EN* and *Language = iso3166-1a2:FR*, then the Creation is in English AND in French (but not in any other language).

However, Attributes can also be combined using an inclusive Boolean disjunction (OR), an exclusive disjunction (XOR) or a negation (NOT), or with equivalent logical constructs in another formalism[6]. For example, a Right may be granted for viewing or editing a Creation (OR), a Party can be located in France or in Germany (XOR), or a Right may apply everywhere except from France (i.e. in the World, but NOT in France).

If multiple Attributes (or values of an Attribute) are not all required, then the Attributes must be explicitly related by one of the Boolean operators OR (if there are multiple values), XOR (if only one of a set of values is required) or NOT (if a value is to be excluded).

Note that a value cannot be excluded without specifying the inclusion of the universe from which it is being excluded: for example, if a Right is valid anywhere in the world except Australia, then the World must be included as a Territory, and Australia excluded ("the World but NOT Australia").

In principle, an Entity may include any level of "nesting" of these values (for example, a user may use *ToolType A* OR (*ToolType B* AND *ToolType C*) but NOT *ToolType C1* (which is a type of *ToolType C* and so needs to be specifically excluded from the previous condition).

[6] Boolean operators are common ways of representing these common logical relations, but they may be expressed in other ways according to the implementation method. They are referred to here in terms of Booleans for simplicity.

Any formal schema using the Entity Model (such as the CRF) needs to accommodate these operators or their equivalent.

2.1.6 **UML diagram**

Figure 2 is a UML diagram of the LCC Entity Model, which is available separately as a .JPG file (*The LCC Entity Model*) and in an .EAP file (*The LCC Rights Reference Model*):

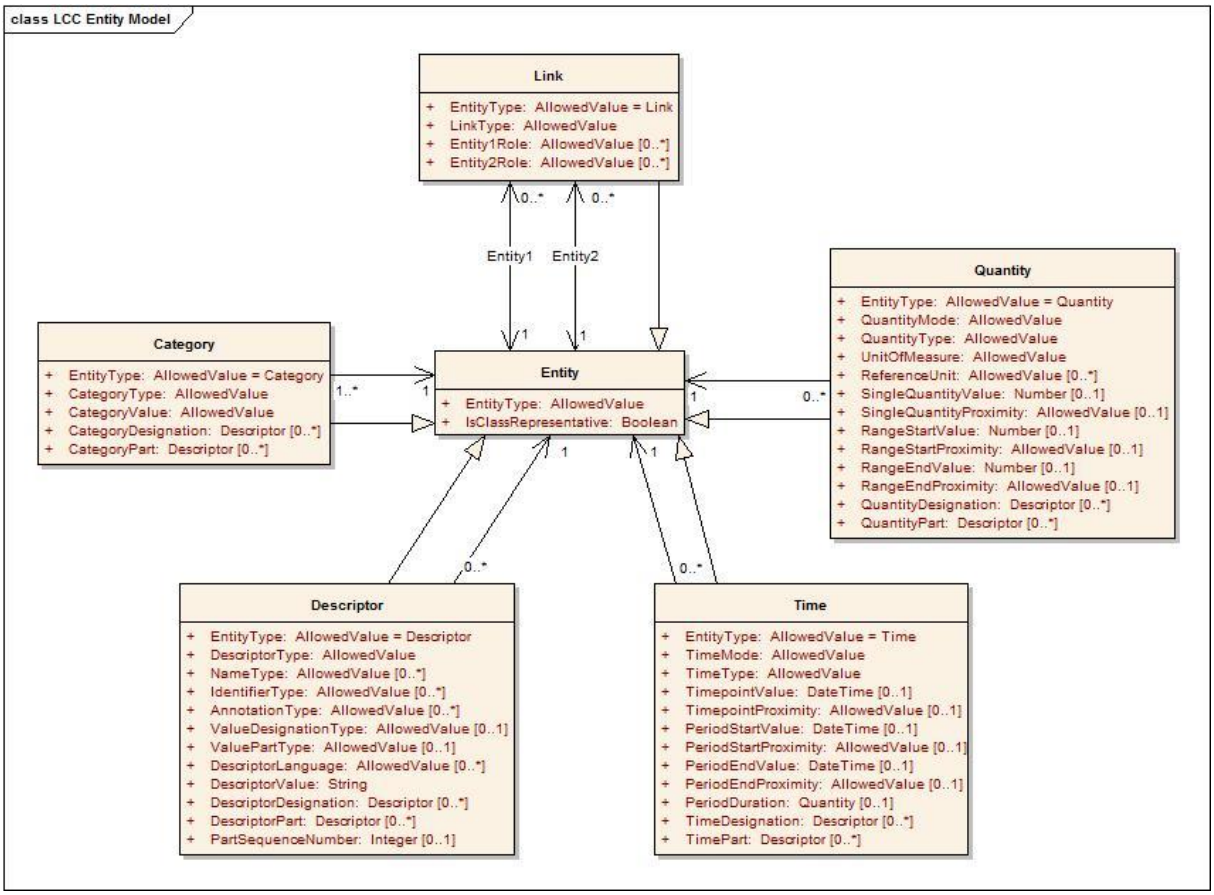


Figure 2: LCC Entity Model, UML class diagram

2.1.7 **XML schema**

The complete LCC Entity Model is represented in the XML Schema **LCC_Entity_Model.xsd.** XML examples in this document conform to this schema.

2.2 Category

An *lcc:Category* Attribute is a fully controlled data value denoting a classification, role or association of an Entity (for example*, lcc:RightType = xyz:Play*). A CategoryValue may be any term from any code list, taxonomy or controlled vocabulary. A CategoryValue is, in effect, a globally unique identifier of a concept.

Categories are the basic building blocks of the LCC Entity Model as they are the means by which Entity and Attribute Types are primarily differentiated. The other four types of Attribute each contain at least one CategoryValue. A Category has the elements shown in Table 3, and as an Entity in its own right may have any number of other Attributes, as may be defined by rules applying to its particular CategoryType.

Table 3: Logical model of a full Category

Element	*Description*	*occ*	*Example, comment*
CategoryOwner	The Entity which owns the Category.	1	
CategoryType	The primary type of the Category.	1	eg *lcc:PartyType*
CategoryValue	The value of the Category taken from a controlled vocabulary. A CategoryValue is always a pre-defined term and is selected from an **Allowed Value Set (AVS)** containing the applicable values. The identity of the AVS applicable to any Category is determined by conditional rules. A Category may have more than one AVS to meet different conditions.	1	eg *lcc:Individual*
CategoryDesignation	A Descriptor (with *DescriptorType = lcc:ValueDesignation*) which is a representation of the CategoryValue.	0-n	eg *"IT", "Italy", "Italia".*

2.2.1 Semantics of Categories

A Category is a Link between an Entity and another Entity, where the second is identified by a controlled value pair (the CategoryType and CategoryValue). Categories are where most of the variable and extensible semantics of the LLC Entity Model are located.

The meaning of a Category generally falls into one of two general semantic groups:

- a **class** of which the Entity is a member (for example, *PartyType = lcc:Organization* or *RightType = xyz:Play*); or less commonly
- an **instance** from an enumerated set of individuals (for example, *Territory = iso3166-1:FR* or *Subject =xyz:EuropeanHistory*).

When a CategoryValue represents an instance, the Category represents a convenient optimization of what otherwise would be expressed as a Link between two individual Entities (for example, a Territory for a Right would be represented as a Link between the Right and a Place). Such "instance" Categories are commonly used for ISO codes for Territories, Languages and Currencies among others, and for subject classification systems such as those maintained by the Library of Congress.

2.2.1.1 **Representation of controlled values**

Semantically, it is immaterial whether the identifier of a CategoryValue is a human-intelligible string or a code so long as it is defined and is unique within its vocabulary. A controlled value denotes a concept which may be represented by any number of different names or labels in the same or different domains. For example, the same concept may be called “publisher”, “éditeur” or “B2” in different vocabularies, and one schema may employ several names or labels for the same term (for example, the ISO 639 Language codes have three variations). It may also be the case that the written definitions in a particular vocabulary are less precise or unambiguous than is desirable: however, this is a data quality issue and does not affect the logical model, where each term still denotes a single unique concept, even if it is not wholly clear to users what that concept is meant to be.

2.2.1.2 **Relationships between controlled values**

Interpreting the semantics of Categories frequently depends not only on the meaning of individual CategoryTypes and CategoryValues themselves, but on the relationships between different controlled values: for example, on the position of a controlled value in a hierarchy of values (for example, that a *xyz:MusicVideo* is a kind of *xyz:Video* and a kind of *xyz:MusicalCreation*), or on the mapping between controlled vocabularies from different schemas so that data can be transformed or aggregated (for example, that *xyz:Video* and *abc:AudiovisualRecording* denote the same thing). This is described in more detail in section 4.4, below.

2.2.2 **XML Category Example**

Here are examples of LCC Categories in XML, compliant with the full LCC_Entity_Model.xsd.

(a) Example of a Category without parts or designations:

```
<Category>
   <CategoryType>lcc:RightType</CategoryType>
   <CategoryValue>xyz:Play</CategoryValue>
</Category>
```

(b) Example of a Category with a Time limitation:

```
<Category>
   <CategoryType>xyz:AvailabilityStatus</CategoryType>
   <CategoryValue>xyz:Available</CategoryValue>
   <Time>
      <TimeMode>lcc:Period</TimeMode>
      <TimeType>lcc:ValidPeriod</TimeType>
      <PeriodStartValue>2005-09-09</PeriodStartValue>
      <PeriodStartProximity>lcc:Exactly</PeriodStartProximity>
      <PeriodEndValue>2005-12-14</PeriodEndValue>
      <PeriodEndProximity>lcc:Exactly</PeriodEndProximity>
   </Time>
</Category>
```

An example of the optimization of a Category by flattening is given in 4.3.1.2.

2.3 **Descriptor**

An *lcc:Descriptor* Attribute is a name, identifier or annotation of an Entity in the form of an uncontrolled or partially controlled data value[7]. It has the elements shown in Table 4, and as an Entity in its own right may have any number of other Attributes, as may be defined by rules applying to its particular DescriptorType.

Table 4: Logical model of a full Descriptor

Element	*Description*	*occ*	*Example, comment*
DescriptorOwner	The Entity which owns the Descriptor.	1	
DescriptorType	A controlled value indicating the primary type of the Descriptor, from *DescriptorType_AVS* (*lcc:Name, lcc:Identifier, lcc:Annotation, lcc:ValueDesignation, lcc:ValuePart*).	1	eg *DescriptorType = xyz:Name*
NameType	A controlled value indicating a type of a Name. Optional when the Descriptor has *DescriptorType = lcc:Name,* otherwise null.	0-n	*eg xyz:ReferenceName*
IdentifierType	A controlled value indicating a type of an Identifier. Mandatory when the Descriptor has *DescriptorType = lcc:Identifier,* otherwise null.	0-n	*eg xyz:ISBN*
AnnotationType	A controlled value indicating a type of an Annotation. Optional when the Descriptor has *DescriptorType = lcc:Annotation,* otherwise null.	0-n	*eg xyz:Comment*
ValueDesignation Type	A controlled value indicating the type of a ValueDesignation. Optional when the Descriptor has *DescriptorType = lcc:ValueDesignation,* otherwise null.	0-1	*eg xyz:IndexedForm*
ValuePartType	A controlled value indicating the type of a ValuePart. Optional when the Descriptor has *DescriptorType* = *lcc:ValuePart,* otherwise null.	0-1	*eg xyz:KeyName*
DescriptorLanguage	A controlled value indicating a language in which the DescriptorValue is expressed.	0-n	*eg iso639-1:en*
DescriptorValue	An alphanumeric string representing the default value of the Descriptor.	1	*eg "Tom Brown"*
DescriptorDesignation	A Descriptor (with *DescriptorType = lcc:ValueDesignation*) which is a	0-n	*eg xyz:IndexedDesignation*

[7] A "partially-controlled" data value is one in which some parts or characteristics are pre-defined by datatyping (for example, a type of identifier may have a three-character alphabetic prefix, a six digit number and a calculated check digit) or may have fixed elements (for example, an ISBN may be prefixed by the characters "ISBN").

	representation of the DescriptorValue.		*= "Brown, Tom"* *xyz:NormalDesignation = "Tom Brown"*
DescriptorPart	A Descriptor (with *DescriptorType = lcc:ValuePart*) which represents a part of the DescriptorValue. DescriptorParts may be assembled using rules to create alternative DescriptorDesignations.	0-n	*eg xyz:NamesBeforeKeyName ="Tom"* *xyz:KeyName="Brown"*
PartSequenceNumber	The default sequence number of a DescriptorPart within a set of DescriptorParts making up one DescriptorValue. Optional when *DescriptorType = lcc:ValuePart*, otherwise null.	0-1	

2.3.1 **Allowed Value Sets for Descriptors**

DescriptorType_AVS is a closed set of terms with the following allowed values:

lcc:Name	A Descriptor which denotes an Entity and which is capable of being written.
lcc:Identifier	A Name which is unique of its type in its domain.
lcc:Annotation	A Descriptor which provides descriptive or other information or opinion about an Entity.
lcc:ValueDesignation	A Descriptor whose function is as a representation of the value of an Attribute (for example, *"Brown, Tom*" as an alternative representation of *"Tom Brown*").
lcc:ValuePart	A Descriptor whose function is as a part of the value of an Attribute (for example, "*Tom*" and *"Brown*" as part of "*Tom Brown*").

lcc:Name is the parent[8] of lcc:Identifier, but the distinction is of such importance that both are shown at this level, and it is normally assumed that any lcc:Name is not intended to be a unique lcc:Identifier. Each DescriptorType may have any number of additional Attribute constraints defined by rules.

2.3.2 **ValueDesignation**

An *lcc:ValueDesignation* is a type of Descriptor whose role is to enable Attribute values to be represented in different ways. For example, the Name *"Tom Brown"* might be represented as *"Tom Brown", "Brown, Tom"* or *"Mr Tom Brown"* in different contexts, while still being considered as the same name.

In principle all of the value-based Attributes (that is, all Attribute Types apart from Link) may have different ValueDesignations. For example, a Territory Category may be represented as *"IT"* or *"Italy*"; a Quantity as *"3"* or *"Three"* and a Time as *"2006-01-01"* or *"1.1.2006".* Any Attribute may have any number of ValueDesignations.

[8] Superclass

Commonly, ValueDesignations may be created by rules applied to ValueParts (see 2.3.3), other elements of the Attribute, or fixed components. For example, the ValueDesignation *"3 cm*" for a Quantity may be derived automatically from the SingleQuantityValue (*"3")* + a space + the CategoryValue of the UnitOfMeasure ("*cm*")

Like all Descriptors, ValueDesignations have a primary type (a ValueDesignationType or some specialization of that): for example, *"Brown, Tom"* might be typed as an *xyz:IndexedForm*, or *"Mr Tom Brown"* as an *xyz:FullNameWithTitle.*

The defined Values shown for an Attribute (DescriptorValue, CategoryValue, TimepointValue, RangeStartValue etc) are the default ValueDesignations. In the majority of real use cases, only these values are provided or needed.

2.3.3 **ValuePart**

An *lcc:ValuePart* is a type of Descriptor which represents a part of a DescriptorValue. For example, the Descriptors "Tom" and "Brown" may be ValueParts of the complete Descriptor "*Tom Brown*".

Like all Descriptors, ValueParts have a primary type (a ValuePartType or some specialization of that): for example (following the ONIX message standard), "*Brown*" might be a *KeyName* and "*Tom*" a *NamesBeforeKeyName.*

The main purpose of ValueParts is for a ValueDesignation to be constructed automatically using a rule. For example, the ValueDesignation *xyz:IndexedForm* may be created by combining the *xyz:KeyName* and *xyz:NamesBeforeKeyName* in that order, inserting a comma after the *xyz:KeyName* ("*Brown, Tom*").

2.3.4 **XML Descriptor Examples**

Here are examples of LCC Descriptors in XML, compliant with the full LCC_Entity_Model.xsd.

(a) Example of a Descriptor (an Identifier) without parts or designations:

```
<Descriptor>
   <DescriptorType>lcc:Identifier</DescriptorType>
   <DescriptorValue>ABC12345</DescriptorValue>
   <Category>
      <CategoryType>lcc:IdentifierType</CategoryType>
      <CategoryValue>xyz:AbcSystemID</CategoryValue>
    </Category>
</Descriptor>
```

(b) Example of a Descriptor (a Name) with parts and designations:

```
<Descriptor>
   <DescriptorType>lcc:Name</DescriptorType>
   <DescriptorValue>Tom Brown</DescriptorValue>
   <Category>
      <CategoryType>lcc:NameType</CategoryType>
      <CategoryValue>xyz:ReferenceName</CategoryValue>
   </Category>
   <Descriptor>
      <DescriptorType>lcc:ValuePart</DescriptorType>
      <DescriptorValue>Tom</DescriptorValue>
```

```
            <Category>
               <CategoryType>xyz:NamePartType</CategoryType>
               <CategoryValue>xyz:NamesBeforeKeyName</CategoryValue>
            </Category>
         </Descriptor>
         <Descriptor>
            <DescriptorType>lcc:ValuePart</DescriptorType>
            <DescriptorValue>Brown</DescriptorValue>
            <Category>
               <CategoryType>xyz:NamePartType</CategoryType>
               <CategoryValue>xyz:KeyName></CategoryValue>
            </Category>
         </Descriptor>
         <Descriptor>
            <DescriptorType>lcc:ValueDesignation</DescriptorType>
            <DescriptorValue>Brown, Tom</DescriptorValue>
            <Category>
               <CategoryType>xyz:NameDesignationType</CategoryType>
               <CategoryValue>xyz:IndexedForm</CategoryValue>
            </Category>
         </Descriptor>
      </Descriptor>
```

An example of the optimization of a Descriptor by flattening is given in 4.3.1.3.

2.4 Quantity

An *lcc:Quantity* Attribute is a numeric measurement of some aspect of an Entity. It has the elements shown in Table 5, and as an Entity in its own right may have any number of other Attributes, as may be defined by rules applying to its particular QuantityType.

Table 5: Logical model of a full Quantity

Element	*Description*	*occ*	*Example, comment*
QuantityOwner	The Entity which owns the Quantity.	1	
QuantityMode	A controlled value identifying whether the Quantity is a SingleQuantity or a QuantityRange, with a value from *QuantityMode_AVS.*	1	eg *lcc:SingleQuantity*
QuantityType	A controlled value identifying the aspect of the Entity which the Quantity is measuring.	1	eg *xyz:NumberOfPages, xyz:DealerPrice, xyz:Duration, xyz:Height*
UnitOfMeasure	A controlled value identifying the unit of measure of the Quantity.	1	eg *lcc:Unit, xyz:cm, xyz:Pixel, xyz:GB, xyz:Dollar, xyz:HHMMSS*
ReferenceUnit	A controlled value identifying the reference unit or denominator of the unit of measure if the Quantity is a ratio. In principle a Quantity can have any number of ReferenceUnits (eg *xyz:Price* per *xyz:Unit* per *xyz:Day*) .	0-n	eg *xyz:Squarecm* if the complete unit of measure is “pixels per square cm”.
SingleQuantityValue	The value of a SingleQuantity. Optional when *QuantityMode* = *lcc:SingleQuantity*, null otherwise.	0-1	eg *“45”*
SingleQuantity Proximity	A controlled value which defines the relationship between the actual SingleQuantityValue and the value that is reported. Mandatory for a Quantity with a SingleQuantityValue, null otherwise, with a value from *QuantityProximity_AVS.*	0-1	eg *lcc:Exactly, lcc:Approximately, lcc:LessThan, lcc:NotMoreThan*
RangeStartValue	The lowest value in a range. Optional when *QuantityMode* = *lcc:QuantityRang*e, null otherwise.	0-1	eg *“1.5”*
RangeStartProximity	A controlled value which defines the relationship between the actual RangeStartValue and the value that is reported. Mandatory for a Quantity with a RangeStartValue, null otherwise, with a value from *QuantityProximity_AVS.*	0-1	eg *lcc:Exactly, lcc:Approximately, lcc:LessThan, lcc:NotMoreThan*
RangeEndValue	The highest value in a range. Optional when *Quantity Mode =*	0-1	eg *“2.5”*

	lcc:QuantityRange, null otherwise.		
RangeEndProximity	A controlled value which defines the relationship between the actual RangeEndValue and the value that is reported. Mandatory for a Quantity with a RangeEndValue, null otherwise, with a value from *QuantityProximity_AVS.*	0-1	eg *lcc:Exactly, lcc:Approximately, lcc:LessThan, lcc:NotMoreThan*
QuantityDesignation	A Descriptor (with *DescriptorType = lcc:ValueDesignation*) which is a representation of the Quantity.	0-n	eg *xyz:NormalDesignation = "Less than 150 grams"*

2.4.1 **Allowed Value Sets for Quantities**

QuantityMode_AVS is a closed set of terms with the following allowed values:

lcc:SingleQuantity	A Quantity which is represented as a single value.
lcc:QuantityRange	A Quantity which is represented as a continuous range between two values.

QuantityProximity_AVS is a closed set of terms with the following allowed values:

lcc:Exactly	Of a Quantity value which is exactly as reported.
lcc:Approximately	Of a Quantity value which is approximately as reported.
lcc:GreaterThan	Of a Quantity value which is more than that reported.
lcc:LessThan	Of a Quantity value which is less than that reported.
lcc:NotGreaterThan	Of a Quantity value which is less than, or the same as, that reported.
lcc:NotLessThan	Of a Quantity value which is more than, or the same as, that reported.
lcc:UnequalTo	Of a Quantity value which not the same as that reported.

Allowed Values for *QuantityType, UnitOfMeasure* and *Reference Unit* may be user-defined according to context. For example, if the Quantity belongs to a Creation which is an e-Book, the AVSs may contain values such as *xyz:Size* or *xyz:NumberOfChapters (from a QuantityType_AVS)* and *xyz:MB* and *lcc:Unit (from a UnitOfMeasure_AVS).*

2.4.2 **XML Quantity Examples**

Here are examples of LCC Quantities in XML, compliant with the full LCC_Entity_Model.xsd.

(a) Example of SingleQuantity without parts or designations:

```
<Quantity>
   <QuantityMode>lcc:SingleQuantity</QuantityMode>
   <QuantityType>lcc:PercentageShare</QuantityType>
   <UnitOfMeasure>lcc:Percent</UnitOfMeasure>
   <SingleQuantityValue>100</SingleQuantityValue>
   <SingleQuantityProximity>lcc:Exactly</SingleQuantityProximity>
</Quantity>
```

(b) Example of a QuantityRange without parts or designations:

```
<Quantity>
   <QuantityMode>lcc:QuantityRange</QuantityMode>
   <QuantityType>xyz:DealerPrice</QuantityType>
   <UnitOfMeasure>xyz:USDollar</UnitOfMeasure>
   <QuantityRangeStartValue>2.50</QuantityRangeStartValue>
   <QuantityRangeStartProximity>lcc:Exactly</QuantityRangeStartProximity>
   <QuantityRangeEndValue>6.50</QuantityRangeEndValue>
   <QuantityRangeEndProximity>lcc:Exactly</QuantityRangeEndProximity>
</Quantity>
```

An example of the optimization of Quantities by flattening is given in 4.3.1.4.

2.5 **Time**

An *lcc:Time* Attribute is a point or period of time associated with an Entity. It has the mandatory elements shown in Table 6, and as an Entity in its own right may have any number of other Attributes, defined by rules applying to its particular TimeType.

Table 6: Logical model of a full Time

Element	*Description*	*occ*	*Example, comment*
TimeOwner	The Entity which owns the Time.	1	
TimeMode	A controlled value identifying whether the Time is a Timepoint or a Period, with a value from *TimeMode_AVS.*	1	eg *lcc:Timepoint*
TimeType	A controlled value identifying the aspect of the Entity which the Time is describing.	1	eg *lcc:DateOfBirth, lcc:ValidPeriod*
TimepointValue	The value of a Timepoint. Optional when *TimeMode = lcc:Timepoint*, null otherwise.	0-1	eg *"2012-01-01"*
TimepointProximity	A controlled value which defines the relationship between the actual TimepointValue and the value that is reported. Mandatory for a Time with a TimepointValue, null otherwise, with a value from *TimeProximity_AVS.*	0-1	eg *lcc:Exactly, lcc:Circa, lcc:After, lcc:NotBefore*
PeriodStartValue	The earliest value in a Period. Optional when *TimeMode = lcc:Period,* null otherwise.	0-1	eg *"2012-01-01"*
PeriodStartProximity	A controlled value which defines the relationship between the actual PeriodStartValue and the value that is reported. Mandatory for a Time with a PeriodStartValue, null otherwise, with a value from *TimeProximity_AVS*	0-1	eg *lcc:Exactly, lcc:Circa, lcc:After, lcc:NotBefore*
PeriodEndValue	The latest value in a Period. Optional when *TimeMode = lcc:Period,* null otherwise.	0-1	eg *"2012-12-31"*
PeriodEndProximity	A controlled value which defines the relationship between the actual PeriodEndValue and the value that is reported. Mandatory for a Time with a PeriodEndValue, null otherwise, with a value from *TimeProximity_AVS*	0-1	eg *lcc:Exactly, lcc:Circa, lcc:After, lcc:NotBefore*
PeriodDuration	A Quantity which defines the duration of the Time, if it is a Period. A PeriodDuration may be used instead of PeriodStartValue and PeriodEndValue where the length of a Period is known, but not its start and	0-1	eg *"3 months"*

	end time (for example, in a Right that is effective for a specified term but does not begin until some Condition of the RightsAssignment, such as payment, has been met). Optional when a Time has a *TimeMode = lcc:Period* with no fixed start or end time, otherwise null.		
TimeDesignation	A Descriptor (with *DescriptorType = lcc:ValueDesignation*) which is a representation of the Time.	0-n	eg "*Not before May 15, 2013"*

2.5.1 **Allowed Value Sets for Times**

TimeMode_AVS is a closed set of terms with the following allowed values:

lcc:Timepoint	A Time which is represented as a single point in time.
lcc:Period	A Time which is represented as a range between two other Times (with a measurable Duration, unlike a Timepoint).

TimeProximity_AVS is a closed set of terms with the following allowed values:

lcc:Exactly	Of a Time value which is exactly as reported.
lcc:During	Of a Time value which is within that reported.
lcc:Circa	Of a Time value which is approximately as reported.
lcc:Before	Of a Time value which is earlier than that reported.
lcc:After	Of a Time value which is later than that reported.
lcc:NotBefore	Of a Time value which is later than, or the same as, that reported.
lcc:NotAfter	Of a Time value which is earlier than, or the same as, that reported.
lcc:UnequalTo	Of a Time value which not the same as that reported.

Allowed Values for *TimeType* may be user-defined according to context.

2.5.2 **XML Time Examples**

Here are examples of LCC Times in XML, compliant with the full LCC_Entity_Model.xsd.

(a) Example of a Timepoint without parts or designations:

```
<Time>
   <TimeMode>lcc:Timepoint</TimeMode>
   <TimeType>lcc:DateOfBirth</TimeType>
   <TimepointValue>1933</TimepointValue>
   <TimepointProximity>lcc:Exactly</TimepointProximity>
</Time>
```

(b) Example of a Period without parts or designations:

```
<Time>
   <TimeMode>lcc:Period</TimeMode>
```

```
    <TimeType>lcc:ValidPeriod</TimeType>
    <PeriodStartValue>2005-09-09</PeriodStartValue>
    <PeriodStartProximity>lcc:Exactly</PeriodStartProximity>
    <PeriodEndValue>2008-09-08</PeriodEndValue>
    <PeriodEndProximity>lcc:NotBefore</PeriodEndProximity>
</Time>
```

An example of the optimization of a Time by flattening is given in Section 4.3.1.5.

2.6 **Link**

A *lcc:Link* Attribute is a typed relationship between two independent Entities. A Link has the elements shown in Table 7, and as an Entity in its own right may have any number of other Attributes, as may be defined by rules applying to its particular LinkType.

Table 7: Logical model of a Link.

Element	*Description*	*occ*	*Example, comment*
LinkType	A controlled value which identifies the type of the Link.	1	eg *lcc:Creation_Party*
Entity1	The first linked Entity. This may be identified by an Identifier (for example, a database primary key or a URI) or by some other means (such as an XML nesting) according to the formal conventions being used.	1	
Entity1Role	A controlled value identifying a Role played in the Link by Entity1. This is expressed as a CategoryValue from an AVS.	0-n	
Entity2	The second linked Entity. This may be identified by an Identifier (for example, a database primary key or a URI) or by some other means (such as an XML nesting) according to the formal conventions being used.	1	
Entity2Role	A controlled value identifying a Role played in the Link by Entity2. This is expressed as a CategoryValue from an AVS.	0-n	eg *xyz:Author*

2.6.1 **Allowed Value Sets for Links**

Allowed Values Sets of LinkType, Entity1Role and Entity2Role are user-defined according to context. For example, if the Link is between a Creation and a Party, it may require LinkType values such as *lcc:Creation_Party*, using Roles such as *dc:Creator* or *xyz:Author*.

Because the LCC Entity Model does not mandate any specific values of EntityType, there are also no mandated values for LinkTypes. The RRM contains a LinkType_AVS including a set of generic LinkTypes for four main EntityType (Creation, Party, Context and Place) which would be applicable to a large number of profiles of the LCC Entity Model.

2.6.2 **XML Link Example**

Here is an example of an LCC Link in XML, compliant with the full LCC_Entity_Model.xsd.

```
<Link>
   <LinkType>lcc:Right_Party</LinkType>
   <Entity2>P1</Entity2>
```

```
    <Entity2Role>lcc:Rightsholder</Entity2Role>
</Link>
```

An example of the optimization of a Link by flattening is given in Section 4.3.1.6.

2.7 Entity Model Profiles

The LCC Entity Model described above supports the modelling of Entities and their relationships in a predictable and extensible way with inherent semantics that go beyond the simple restrictions of datatypes.

The LCC Rights Reference Model is the first example of a **Profile** of the LCC Entity Model. In a Profile of the LCC Entity Model, specific types of Entity are defined by rules which constrain the number and type of Attributes which each has, based normally on the unambiguous values of Categories and (less commonly) other Attributes. For example, these additional constraints are two of those which apply in the RRM:

> An Entity with *lcc:EntityType = lcc:Place* has exactly one Category with *lcc:CategoryType = lcc:PlaceType* with a value from *lcc:PlaceType_AVS.*
>
> An Entity with *lcc:EntityType = lcc:Right* has at least one Link of type *lcc:Right_Party* with an *lcc:Entity2Role* of *lcc:Rightsholder,* where Entity2 has *lcc:EntityType = lcc:Party,* and there are no values of *lcc:Entity1Role.*

Constraint rules may be expressed in any suitable schema, programming or rules language. In section 4 of the LCC Rights Reference Model, constraints are expressed in abstract terms in a tabular form, and in a UML class diagram.

In an LCC Entity Model Profile there may be any number of optimizations in terms of flattened or displaced Attributes, as described in Section 4.

3 Terms and Allowed Value Sets (AVSs)

This section lists the Allowed Value Sets (AVSs) and values within them mandated by the LCC Entity Model. Other values and AVSs may be added by users as required with an Entity Model Profile such as the RRM.

Term	Definition[9]	Comments, examples
lcc:AttributeType_AVS	A closed Allowed Value Set in which each value is a type of Attribute.	
lcc:Category	A categorization of an Entity with a fully-controlled data value.	*RightType = xyz:Copy* *Language = iso3166-1a2:EN ("English")* *WorkType = xyz:AudiovisualWork*
lcc:Descriptor	A name, identifier or annotation of an Entity in the form of an uncontrolled or partially controlled data value.	*Name = "Tom W Brown"* *ISBN = "9740946014632"* *Note = "Do not open until Christmas"*
lcc:Link	A typed relationship between two Entities.	"Moby Dick" has the Author "Herman Melville", modelled as a Link between two Entities representing "Moby Dick" and "Herman Melville" with *LinkType = lcc:Creation_Party* and *Entity2Role = xyz:Author.*
lcc:Quantity	A measure of some aspect of an Entity.	*PercentageShare = 50%* *NumberOfUses = 25* *Height = 22 cm*
lcc:Time	A point or period of time associated with an Entity.	*PeriodStartTime = 2013-01-03* *DateOfCreation = 1975*
lcc:DescriptorType_AVS	A closed Allowed Value Set in which each value is a type of Descriptor.	
lcc:Name	A Descriptor which denotes an Entity and which is capable of being written.	
lcc:Identifier	A Name which is unique of its type in its domain.	
lcc:Annotation	A Descriptor which provides descriptive or other information or opinion about an Entity.	
lcc:ValueDesignation	A Descriptor of an Attribute, providing a representation of an AttributeValue.	For example, *"Brown, Tom*" as an alternative representation of *"Tom Brown*".
lcc:ValuePart	A Descriptor of an Attribute, providing a	For example, "*Tom*" and *"Brown*"

[9] Note that the definitions of Allowed Values may be specialized for the AVS to which they belong: for example, the definition of "lcc:Tool" in the AVS for lcc:Right_Creation Roles limits a Tool there to being a Creation, whereas other types of Entity (such as Contexts and Parties) may play the role of a Tool elsewhere.

	representation of part of an AttributeValue.	as part of "*Tom Brown*").
lcc:QuantityMode_AVS	A closed Allowed Value Set in which each value is a mode of Quantity.	
lcc:SingleQuantity	A Quantity which is represented as a single value.	
lcc:QuantityRange	A Quantity which is represented as a continuous range between two values.	
lcc:QuantityProximity_AVS	A closed Allowed Value Set in which each value represents the relationship between the reported value of a Quantity and the actual value of the Quantity it describes.	
lcc:Exactly	Of a Quantity value which is exactly as reported.	
lcc:Approximately	Of a Quantity value which is approximately as reported.	
lcc:GreaterThan	Of a Quantity value which is more than that reported.	
lcc:LessThan	Of a Quantity value which is less than that reported.	
lcc:NotGreaterThan	Of a Quantity value which is less than, or the same as, that reported.	
lcc:NotLessThan	Of a Quantity value which is more than, or the same as, that reported.	
lcc:UnequalTo	Of a Quantity value which not the same as that reported.	
lcc:TimeMode_AVS	A closed Allowed Value Set in which each value is a mode of Time.	
lcc:Timepoint	A Time which is represented as a single point in time.	
lcc:Period	A Time which is represented as a range between two other Times (with a measurable Duration, unlike a Timepoint).	
lcc:TimeProximity_AVS	A closed Allowed Value Set in which each value represents the relationship between the reported value of a Time and the actual value of the Time it describes.	
lcc:Exactly	Of a Time value which is exactly as reported.	
lcc:During	Of a Time value which is within that reported.	
lcc:Circa	Of a Time value which is approximately as reported.	
lcc:Before	Of a Time value which is earlier than that reported.	
lcc:After	Of a Time value which is later than that	

	reported.	
lcc:NotBefore	Of a Time value which is later than, or the same as, that reported.	
lcc:NotAfter	Of a Time value which is earlier than, or the same as, that reported.	
lcc:UnequalTo	Of a Time value which not the same as that reported.	
lcc:UnitOfMeasure_AVS	An open Allowed Value Set in which each value is a unit of measure of a Quantity.	Other values of this AVS may be user-defined.
lcc:Percent	A UnitOfMeasure of a Percentage which represents 1%.	
lcc:Unit	A UnitOfMeasure which represents a single counted Entity.	

4 Use and optimization

4.1 Strengths and weaknesses of the full Entity model

Here are four main benefits of the full LCC Entity Model:

- it is **comprehensive** to support all types of Attribute for any kind of Entity within the domains and models covered by the scope of the LCC.[10]
- it is **extensible** to allow for new and currently unknown types of Entity and Attribute and business rules without the need for changing the existing schema, only adding new EntityTypes, vocabularies and rules (so a system can be heavily parameterized and "data-driven").
- it is a very **rich** model, so most other schemas should map successfully to it.
- each Attribute of every Entity Type conforms to one of the five micro-model structures, making it possible to write **generic, modular software** which can be used for processing metadata about any Entities, regardless of future extensions.

Against this there is one significant disadvantage: the full model contains a high level of indirection which means that data processing may be inefficient, and hence optimizations will be useful or essential in different situations.

4.2 Use of the model

There are any ways in which the Entity Model may be used, five of which are briefly described here. In the first (**data transformation**) and fifth (**quality assurance**), the model is best used in its full form; in the second (**large-scale modelling**) it may be selectively optimized, and in the remaining two (**small-scale modelling** and **messaging**) the model will typically be quite extensively optimized.

4.2.1 Data integration and transformation

A Profile of the Entity Model can provide a model for a generic database schema to act as an intermediate stage in data transformation from any schema to any other. This may be in support of one-to-one data exchanges between two different schemas or silos, or more usefully in a "hub-and-spoke" situation where several different data schemas need to be integrated with each other in different ways[11].

Data can be aggregated in the intermediary stage, so that multiple sources can be consolidated into a single schema before further use is made of it.

4.2.2 Large scale modelling and design

[10] In principle the LCC Entity Model can be applied to any kind of Entity in any domain, but there are domains in which some of the underlying assumptions of the model may not apply (for example, mathematical or scientific databases with non-linear views of time). There are also types of Entity - such as financial transactions - which could be supported by the Entity Model but for which other modelling techniques will be more established and efficient. It is particularly well suited for domains such as rights which are heavily dependent on the use of Categories, Times and Descriptors rather than Quantities.

[11] This use of the Entity Model is what will be extensively tested in the Rights Data Integration (RDI) Project.

A Profile of the Entity Model provides a "rich as possible" abstract data model which can be translated directly into a physical model to provide the basis of any extensive, multi-media rights management system or network in which large-scale data processing is not a seriously limiting factor.

4.2.3 Smaller scale modelling

On a smaller scale, the Entity Model (or a Profile of it) provides a starting point from which a system can be modelled. The key is that the modelling process enables a designer to start from the full scale model, and then specialize and optimize to a flatter and more "displaced" schema (see section 4.3 for a summary of the two main optimization techniques). It should be possible then to "enrich" the resulting system relatively easily if needed in future by reversing the optimization steps.

4.2.4 Messaging

The process for designing messages is parallel to that of smaller scale modelling. Starting from a generic schema (such as the CRF), messages can be created by the selection and optimization of required elements.

4.2.5 Quality Assurance

The Entity Model, or a Profile of it, can be used as a template for QA-ing existing or proposed models and schemas for completeness.

4.3 Optimization of the model

There are two main ways in which the Entity Model can be optimized for performance as a physical data model: Flattened Attributes (see 4.3.1) and Displaced Attributes (see 4.3.2).

4.3.1 Flattened Attributes

In the full form of the LCC Entity Model, every Attribute is made up of at least two data elements, and is also an Entity in its own right which may have further Attributes of its own. For example, if (in the RRM Profile of the Entity Model) a Party requires a DateOfBirth element in a particular system, the Profile would model this as a Time with these elements:

AttributeType	lcc:Time
TimeMode	lcc:Timepoint
TimeType	lcc:DateOfBirth
TimepointProximity	lcc:Exactly
TimepointValue	1953

whereas in many schemas this would be expressed as a simple type-value pair:

DateOfBirth	1953

This second example is described as a "Flattened Attribute" in the LCC Entity Model. Within any system an Attribute may be flattened to a single type-value pair under specific conditions.

The first of these conditions is common to all five LCC Attribute Types:

- *No instance of the specialised Attribute Type ever requires Attributes of its own in the system under consideration*

For example, if the TimeType "DateOfBirth" in the system under consideration *may* sometimes require a related comment (an lcc:Descriptor), or a status indicating the level of reliability attached to its value (an lcc:Category), or a link to the Party on whose authority this particular Attribute was created (an lcc:Link), or some other Attribute, then it must remain an Entity in its own right and cannot be flattened.

If this is not the case, each type of Attribute has different additional rules for flattening, but the principle is that:

- *Every instance of the Attribute Type has the same default values for supporting elements in the system under consideration*

In the DateOfBirth example given above, this means that:

- *A DateOfBirth is always expressed as an lcc:Timepoint (never as an lcc:Period)*
- *The TimepointProximity value is always lcc:Exactly (not, for example, lcc:Circa)*

If either of these conditions are not always true (in a bibliographic system, for example, it is likely that the second will not be) then the element cannot be flattened. The same principle applies to all five Attribute types.

The principle is that, so long as the flattened Attribute can be *automatically and unambiguously transformed into a full Attribute by rules*, then it is compatible with the full RRM. If it cannot be, then it is almost certain that there is a modelling error in the schema in which the Attribute appears, regardless of any mapping to the RRM (this is an example of how the RRM can be used to QA an existing model).

4.3.1.1 LCC Entity Model Flattened Attribute models

Adopting this standard approach to flattening Attributes means in effect that the LCC Entity Model has ten AttributeTypes instead of five - five "full" AttributeType and five "flattened" AttributeTypes, each of which has its own set of rules. This increases the potential complexity of data processing, as there are now two models for each AttributeType, and any processes which cross examples of both must be programmed to manage both: as with all data structure optimizations, there is a trade-off of speed against expressiveness.

Other Attribute flattening rules are as follows:

Attribute Type	Flattening rules: the Attribute may be flattened if...
Category	1. No instance of the CategoryType ever requires an Attribute of its own.
	2. No instance of the CategoryType ever requires an lcc:ValueDesignation.
Descriptor	*Note: The "DescriptorType" referred to in the rules actually denotes the specialized NameType, IdentifierType or AnnotationType, according to whatever the general DescriptorType is.*
	1. No instance of the DescriptorType ever requires an Attribute of its own.
	2. No instance of the DescriptorType ever requires an lcc:ValueDesignation.
	3. No instance of the DescriptorType ever requires an

	lcc:ValuePart.
	4. If the DescriptorType has one or more lcc:Languages, every instance has the same value.
Quantity	1. No instance of the QuantityType ever requires an Attribute of its own.
	2. No instance of the QuantityType ever requires an lcc:ValueDesignation.
	3. Every instance of the QuantityType has the same value of QuantityMode (whether lcc:SingleQuantity or lcc:QuantityRange).
	4. Every instance of the QuantityType has the same value for each Proximity of the same type.
	5. Every instance of the QuantityType has the same value of lcc:UnitOfMeasure.
	6. Every instance of the QuantityType has the same value of lcc:ReferenceUnit (if any).
	7. If the QuantityMode is always lcc:QuantityRange, the flattened element value is expressed in a standard way as a single value.
Time	1. No instance of the TimeType ever requires an Attribute of its own.
	2. No instance of the TimeType ever requires an lcc:ValueDesignation.
	3. Every instance of the TimeType has the same value of TimeMode (whether lcc:Timepoint or lcc:Period).
	4. Every instance of the TimeType has the same value for all Proximities of the same type.
	5. If the TimeMode is always lcc:Period, the value is never expressed using an lcc:PeriodDuration in any instance of the TimeType.
	6. If the TimeMode is always lcc:Period, the flattened element value is expressed in a standard way as a single value.
Link	1. No instance of the LinkType ever requires an Attribute of its own.
	2 Neither lcc:Entity1Role nor lcc:Entity2Role has any values in any instance of the LinkType; or there is a single value for one of these Roles which is identical for all instances (for example, an *lcc:Creation_Party* Link with lcc:Entity2Role *xyz:Author* may become a single element *AuthorID*).

In a typical LCC Entity Model flattening, the name of the flattened element will typically be based on the name which would otherwise be used as the AttributeType (as in the *DateOfBirth* example above), incorporating other constant sub-element values where necessary (for example, *PriceInDollars*, *EarliestStartTime, Author*).

4.3.1.2 Flattened Category example

Here is an example of an LCC Category in XML, compliant with the full LCC_Entity_Model.xsd.

```
<Category>
   <CategoryType>lcc:RightType</CategoryType>
   <CategoryValue>xyz:Play</CategoryValue>
</Category>
```

If no instance of **lcc:RightType** in the system under consideration ever requires ValueParts, ValueDesignations or additional Attributes, this can be flattened to:

```
<RightType>xyz:Play</RightType>
```

4.3.1.3 Flattened Descriptor example

Here is an example of an LCC Descriptor in XML, compliant with the full LCC_Entity_Model.xsd. In this example, terms with the prefix "xyz:" represent user-specified terms in any namespace.

```
<Descriptor>
   <DescriptorType>lcc:Identifier</DescriptorType>
   <DescriptorValue>ABC12345</DescriptorValue>
   <Category>
      <CategoryType>lcc:IdentifierType</CategoryType>
      <CategoryValue>xyz:AbcSystemID</CategoryValue>
    </Category>
</Descriptor>
```

If no instance of **xyz:AbcSystemID** in the system under consideration ever requires ValueParts, ValueDesignations or additional Attributes, this can be flattened to:

```
   <AbcSystemID>ABC12345</AbcSystemID>
```

4.3.1.4 Flattened Quantity example

Here is an example of an LCC Quantity in XML, compliant with the full LCC_Entity_Model.xsd. In this example, terms with the prefix "xyz:" represent user-specified terms in any namespace.

```
<Quantity>
   <QuantityMode>lcc:SingleQuantity</QuantityMode>
   <QuantityType>lcc:PercentageShare</QuantityType>
   <UnitOfMeasure>lcc:Percent</ UnitOfMeasure>
   <SingleQuantityValue>100</SingleQuantityValue>
   <SingleQuantityProximity>lcc:Exactly </SingleQuantityProximity>
</Quantity>
```

If every instance of **lcc:PercentageShare** in the system under consideration is an lcc:SingleQuantity and exact, is always expressed as a percentage, and never requires additional Attributes, this can be flattened to:

```
<PercentageShare>100</PercentageShare>
```

4.3.1.5 Flattened Time example

Here is an example of an LCC Time in XML, compliant with the full LCC_Entity_Model.xsd. In this example, terms with the prefix "xyz:" represent user-specified terms in any namespace.

```
<Time>
   <TimeMode>lcc:Timepoint</TimeMode>
```

```
    <TimeType>lcc:DateOfBirth</TimeType>
    <TimepointValue>1953</TimepointValue>
    <TimepointProximity>lcc:Exactly</TimepointProximity>
</Time>
```

If every instance of **lcc:DateOfBirth** in the system under consideration is an lcc:SingleQuantity and exact, is always expressed as a year, and never requires additional Attributes, this can be flattened to:

```
<DateOfBirth>1953</DateOfBirth>
```

4.3.1.6 Flattened Link example

Here is an example of an LCC Link in XML, compliant with the full LCC_Entity_Model.xsd. In this example, terms with the prefix "xyz:" represent user-specified terms in any namespace.

```
<Link>
    <LinkType>lcc:Right_Party</LinkType>
    <Entity2>P1</Entity2>
    <Entity2Role>lcc:Rightsholder</Entity2Role>
</Link>
```

If every instance of **lcc:Right_Party** with the Role **lcc:Rightsholder** in the system under consideration never requires additional Attributes, this can be flattened to:

```
<Right_Rightsholder>P1</Right_Rightsholder>
```

4.3.1.7 Variant approaches to flattening

There are other approaches to flattening, two of which are illustrated here using the DateOfBirth example.

One is to use Displaced Attributes (discussed further in 4.3.2). In the example Time given here, if the only variable element apart from the value is the TimepointProximity, then the Party might be given two distinct elements:

DateOfBirth	1953
DateOfBirthProximity	lcc:Exactly

so that the Proximity can be varied when required. In this case the Proximity element is a *Displaced Attribute*, because while it is actually an Attribute of the Time, it is shown in the model as a direct Attribute of the Party (in this case, as a flattened Category). Logically this does not create a problem, as there is no ambiguity (because there can be only one DateOfBirth), but it illustrates how quickly a structural optimization may become a complication for processing: in place of an instance of the generic Time entity which can be queried in the same way as every other Time in the system, the system has now introduced two specialized attributes which require their own query patterns. Whether such trade-offs are worth it will depend on the context.

A second common variant approach is to change the datatype of the Attribute Value: in this case the DateOfBirth value would allow a string such as (say) "c. 1953" or "not before 1953". In terms of the LCC Entity Model, this turns the Time into a Descriptor (an Annotation) and introduces a new dimension of complexity into the processing: the value no longer has consistent behaviour and so querying, listing and calculating based on it becomes unpredictable or impossible. Again, whether such trade-offs are worth it will depend on the context.

4.3.2 **Displaced Attributes**

A Displaced Attribute is an Attribute which properly belongs to one Entity, but is assigned to another, normally to flatten the model.

For example, the name of the author of a book is often shown in metadata as an Attribute of the Book itself:

Title	Moby Dick
Author	Herman Melville

whereas of course "Herman Melville" is the name of the Party who plays the role of author in the book.

In modelling this example with the full RRM, the two Entities would be separately identified, each would be assigned their respective Names, and then a Creation_Party Link established between them with an Entity2Role of "xyz:Author". However, provided that the flattening rules are met with, there is no reason why a flattening of this kind cannot be used in an optimized RRM (note though that whenever a Name or Identifier of one Entity is displaced to another, it becomes an Annotation of the other in RRM).

Displacement can occur to any level - for example, the release version of the format of the filetype in which a Work may be made available might be shown as a direct Attribute (an Annotation) of the Creation, being four Entities displaced. So long as it remains possible to *automatically and unambiguously transform the displaced Attribute into its full RRM form by rules*, then it is compatible with the full RRM. As with other flattened Attributes, if this is not possible, then it is probably evidence of a modelling error in the schema.

4.3.3 **Flattened Sets**

In the full Entity Model, a Set of two or more elements would be treated as an Entity (with a Category such as *xyz:EntityType = xyz:Set*) which then has Links to each of its members (with a LinkType such as *xyz:Set_Member*). This allows members to be ordered, or for their links to the Set to have any other required Attributes (such as the Time of joining the Set, or the Quantity of the member included in the Set, or the exclusion of a particular member whose parent is otherwise included).

For implementations which requires non-ordered Sets with no Attributes on their Links, a flattened Set Entity might be defined which contains a repeating EntityID element for its members.

4.4 **Constraint rules and controlled vocabulary relationships**

As described in section 2.1.4 above, the Entity Model is extended or specialized by using constraint rules to meet the requirements of any particular domain or enterprise. These rules determine the valid types, numbers and combinations of Attributes for any particular type of Entity in a schema or system. For example, to allow Parties to be divided into individuals and organizations:

> "An Entity with *lcc:EntityType = lcc:Party* may have one occurrence of a Category of type *lcc:PartyType*, with values drawn from the controlled vocabulary *lcc:PartyType_AVS*. "

or to allow a Creation to have any number of identifiers of certain types:

> "A Entity with *lcc:EntityType = lcc:Creation* may have any number of Descriptors with *lcc:DescriptorType = lcc:Identifier* where *lcc:IdentifierType* has a value from ONIX code list 5."

Rules like these may be expressed in any appropriate formalism (such as a database schema, a rules language or procedural code), but will invariably reference at least one, and frequently several,

controlled values, as in the examples above. Controlled values within constraint rules are therefore the principal points at which additional semantic information is brought into a particular model from outside of the LCC Entity Model itself.

The definition of controlled vocabularies is entirely a matter for a particular domain, but there are common ways of expressing relationships which exist between controlled values outside of the model environment, which are of value and which can be utilised in constraint and processing rules.

4.4.1 **Hierarchies and ontologies**

The most common logical relationships between controlled values are *hierarchical*, parent-child links (known at times as subclass, subrelator or subtype relationships). Hierarchies support the inheritance or subsumption of Attributes from one concept to another.

Other kinds of logical relationships between controlled values include *disjunction* (whereby an Entity may not belong to both class A and class B), *whole-part* and *class-instance* relationships. All such logical relationships may be stored in tables or in supporting **ontologies** which may be referenced automatically for data processing. Many changes to a particular LCC-based model or system can then be managed through additions to the terms, relationships and rules in a supporting ontology rather than changes to a schema or procedural code.

4.4.2 **Vocabulary mapping**

Logical relationships ("mappings") between controlled values from different schemas, domains or namespaces are essential to support interoperability between systems, so that CategoryValues from one namespace can be converted to corresponding values in another. Mappings may be "*is same as*" relationships or they may be hierarchical or any of the logical relationship types referenced in 4.4.1. These may also be stored in the same ontology as other logical relationships.