Literate Data Model

Preliminaries the basic structureof the model

content: In Literate Data Modeling, the main components of interest are typically Classes, Attributes, Models, and Subjects. However, to streamline the model and promote reusability, we introduce a supertype called Component. By defining common attributes and behaviors in the Component class, we can inherit them in the subclasses, ensuring consistency and reducing duplication throughout the model.

content: We present the Component class first because it is a best practice in modeling to introduce supertypes before their subtypes. This approach allows readers to understand the general concepts and shared properties before delving into the specifics of each specialized component.

_ **Component** An element or building block of the literate data model

- **normalName** the name of the component, not in camel case **String**
- **name** The name of the component *CamelName*
- **qualifiedName** **QualifiedCamel**
- **abbreviatedName** a short form of the component's name, used for cross references and improved readability. **CamelName**

default: **AS_ENTERED:** name

emoji:

label: Example

content: "LDM" is the short form of "Literate Data Model".

- **oneLiner** A brief, one-line definition or description of the component, suitable for use in a descriptive table of contents. _ ***RichLine***

- **elaboration** A more detailed explanation or discussion of the component _ ***RichText***

__ **For Machinery** mechanical attributes

- **isEmbellishment** Indicates whether this component is an embellishment added during post-parsing processing _ ***Boolean***

  default: **AS_ENTERED:** false

  emoji:

  label: Note

  content: This attribute is set to true for components that are automatically generated or added during the fleshing out, review, or rendering processes, such as implied attributes or suggested model elements. It helps distinguish embellishments from the core model elements defined in the original LDM source.

__ **Markdown Support**

- **mdPrefix** ***[String](#string***

  derivation: **AS_ENTERED:** ""

- **mdSuffix** ***[String](#string***

  derivation: **AS_ENTERED:** ""

- **mdTopLine** ***[String](#string***

  derivation: **AS_ENTERED:** mdPrefix + name + " - " + oneLiner + mdSuffix

_ **AnnotationType** a kind of note, or aside, used to call attention to additional information about some Component.

based_on: Literate Data Model

emoji:

label: Note

content: Each LDM declares a set of Annotation Types, with defined labels, emojis, and clearly documented purposes. These are *recognized* or *registered* Annotation Types.But, if none of these fit, you can introduce an Annotation with any label. It would have an *ad hoc* Annotation Type.

- **emoji** an emoji *Emoji*
- **emojiName** an emoji *String*
- **emojiUnicode** the Unicode for the emoji *Unicode*
- **label** A short label to indicate the purpose of the annotation _ *CamelName*
- **plural** the plural form of the label **UpperCamel**

default: AS_ENTERED: based on label

- **Purpose** the intended reason for the annotation.

_ **ValueType: AnnotationA note or comment associated with a model element**

based_on: Component

- **annotationType** *optional Annotation Type*

emoji:

label: Note

content: An Annotation is considered to *recognized* if the label is associated with an Annotation Type. otherwise it is *ad hoc*.

- **label** A short label to indicate the purpose of the annotation _ *CamelName*

  content: But any short label is valid.

  default: **AS_ENTERED:** from annotationType

- **Emoji** *optional Emoji*

  default: **AS_ENTERED:** from annotation type

- **content** The content or body of the annotation **RichText**

__ **For Machinery**

- **isEmbellishment** Indicates whether this annotation is an embellishment added during post-parsing processing _ *Boolean*

  default: **AS_ENTERED:** false

  emoji:

  label: Note

  content: This attribute is set to true for annotations that are automatically generated or added during the fleshing out, review, or rendering processes, such as suggestions, issues, or diagnostic messages. It helps distinguish embellishment annotations from the annotations defined in the original LDM source.

The Model and its Subjects

_ **LiterateDataModel** A representation of a domain's entities, attributes, and relationships,along with explanatory text and examples

abbreviation: LDM

plural: LiterateDataModels

subtype_of: **Component**

- **name** *UpperCamel*

- **allSubjects** list of all classes in the model, as ordered in thedefinition of the model. *List of Classes*

  derivation: **AS_ENTERED:** gathering s.allSubjects over s in subjectAreas

  as_entered: Subject names must be unique across the model.

- **allClasses** list of all classes in the model, as ordered in the definition of the model. *List of Classes*

  derivation: **AS_ENTERED:** gathering s.allClasses over s in allSubjects.

  as_entered: Class names must be unique across the model.

__ **Modeling Configuration**

- **annotationTypes** *List of AnnotationTypes*

- **Preferred Coding Language** the recommended lanquage for expressing derivation, defaults, and constraints *Coding Language*

  default: **AS_ENTERED:** OCL

- **alternate Coding Languages** *optional List of Coding Languages*

- **Preferred Template Language** the recommended lanquage for expressing derivation, defaults, and constraints *Template Language*

  default: **AS_ENTERED:** Handlebars

- *optional List of Template Languages*

alternate Template Languages

- **aiFunctions** A list of functions that require sophisticated AI-powered implementation * ***List of String***

  derivation: **AS_ENTERED:** ['aiEnglishPlural()']

__ **Markdown Support**

- **mdPrefix** ***[String](#string***

  derivation: **AS_ENTERED:** "# "

- **mdTopLine** ***[String](#string***

  derivation: **AS_ENTERED:** mdPrefix + name

_ **SubjectA specific topic or theme within the model**

plural: Subjects
subtype_of: Component
dependent_of: LiterateDataModel
content: Subjects are the chapters an sections of the model. + A subject need not contain any Classes if itâ€™s just expository.

- **name** ***UpperCamel***

- **parentSubject** The parent subject, if any, under which this subject is nested _ ***Subject, optional***

- **Classes** The major classes related to this subject, in the order in which they should be presented _ ***ListOf Classes***

  emoji:
  label: Issue
  content:

define chapter, section, subsection as levels? ***DSL***: Generally, it is best to present the classes within a Subject in top down order:

- **Each Class should be followed first by the classes that are dependent on it, and then**

- **By its subtype classes.**

- **childSubjects** Any child subjects nested under this *ListOf* subject, in the order in which they *Subjects* should be presented _

  inverse: CLASS_NAME: Subject
  ATTRIBUTE_NAME: parentSubject

  content: ***DSL***: the Classes within a Subject are always displayed before the childSubjects.

__ **Markdown Support**

- **mdPrefix** **[String](#string*

  derivation: AS_ENTERED: levelIndicator + " "

- **mdTopLine** **[String](#string*

  derivation: AS_ENTERED: mdPrefix + name.

_ **SubjectAreaA main topic or area of focus within the model, containing related subjects and classes**

plural: SubjectAreas

subtype_of: Subject

where: parentSubject is absent

Classes

_ **Class** A key entity or object type in the model, often corresponding to a real-world concept

plural: Classes

subtype_of: Component

as_entered: Within each Class, attribute names must be unique.

- **pluralForm** the normal English plural form of the name of the Class *UpperName*

  content: Might be Books for the Book class or other regular plurals. + But also might be People for Person.

  emoji:

  label: Note

  content: When inputting a model, you will rarely need to specify the plural form. The input program will just look it up.

  content: The exception is when a common noun has two plural forms, like People and Persons. But this is unusual.

  default: **AS_ENTERED:** the regular plural, formed by adding "s" or "es".

- **basedOn** the Class or Classes on which this class is dependent *SetOf Classes*

  content: This is solely based on **Existence Dependency**. A true dependent entity cannot logically exist without the related parent entity. For instance, an Order Item cannot exist without an Order. If removing the parent entity logically implies removing the dependent entity, then it is a dependent entity.

  emoji:

  label: Note

  content: that basedOn and dependentOf are being used synonymousle in this metamodel.ToDo - fix that

- **supertypes** The parent class *es*

- **subtypings** — the criteria, or dimensions, by which the class can be divided into subtypes — *list of Subtypings*

  emoji:

  label: Example

  content: in a library model, the `Book` class could have subtypings based on genre (e.g., Fiction, Non-fiction), format (e.g., Hardcover, Paperback), or subject (e.g., Science, History).

- **subtypes** — Any subtypes or specializations of this class based on itâ€™s subtypings. _ — *ListOf Classes*

  emoji:

  label: Example

  content: For instance, using the `Book` example, the subtypes could include `FictionBook`, `Non-fictionBook`, `HardcoverBook`, `PaperbackBook`, `ScienceBook`, and `HistoryBook`.

- **attributes** — The attributes or properties of the class, in the order in which they should be presented _ — *ListOf Attributes*

- **attributeSections** — additional attributes or properties of the class, grouped for clarity and elaboration. _ — *ListOf AttributeSections*

- **constraints** — Any constraints, rules, or validations specific to this class _ — *ListOf Constraints*

  emoji:

  label: Note

  content: Constraints may be expressed on either the Class or the Attribute. Always?Add examples where

clarity would favor one or the other. Sometimes just a matter of taste.

- **methods** Any behaviors or operations associated with this class _ *ListOf Methods*

__ **Implied Attributes**

- **dependents** the Classes which are basedOn this Class *optional SetOf Classes*

  inverse: **CLASS_NAME:** Class
  ATTRIBUTE_NAME: basedOn

- **UniqueKeys** *optional Set of UniqueKeys*

  inverse: **CLASS_NAME:** UniqueKey
  ATTRIBUTE_NAME: basedOn

_ **Subtyping** a way in which subtypes of a Class may be classified *Subtype of Component*

dependent_of: Class

- **name** *Upper Name*

- **is exclusive** *Boolean*

  default: **AS_ENTERED:** true

- **is exhaustive** *Boolean*

  default: **AS_ENTERED:** true

- **classes** *List of Classes*

  content: ***DSL***: Shown in the DSL as + > Subbtypes: byBrand - Brand1, Brand2,... (non exclusive, exhaustive) + on the super class. And as + > Subtype of: SuperClass byBrand + on the subclass.

  emoji:

label: Note

content: every class can have an unnamed subtyping.Also, each subtyping is by default Exclusive and Exhaustive. So those stipulations may be omitted.

_ **ValueType**

subtype_of: [Class.]

__ **Markdown Support**

- **mdPrefix** **[String](#string*

derivation: AS_ENTERED: "ValueType: ".

_ **Reference Type**

subtype_of: [Class.]

_ **CodeTypeA data type or enumeration used in the model**

subtype_of: [ValueType.]

emoji:

label: Note

content: Often, a CodeType will be assigned to just one attribute in the model. In such cases, there's no need to declare a new Code Type and invent a name for it. Instead:

- **List the code values as a bulletted list inside the description of the attribute in the form:â€˜code: descriptionâ€™**

- **A Code Type will be created with the name [class] [attribute]Code and the code values listed. That CodeType will be marked as isCaptive.**

- **isCaptive** the code type was implied by use in an attribute and is only used for that attribute *Boolean*

_ **Code Value**

emoji:

label: A possible value for an enumerated data class DependentOf

content: CodeType

- **code** A short code or abbreviationi for the value _ *NameString*

- **description** an explanation of what the code means **RichText**

_ **Key** a list of attributes of a class

subtype_of: Component

dependent_of: Class

- **keyAttributes** the attributes of the base Class. *List of Attributes*

as_entered: each attribute must be a direct or inherited of the base class.

as_entered: no repetitions allowed in keyAttributes> ðŸ‘‍ **Issue**: introduce PureLists?

emoji:

label: Issue

content: need ascending descending to support index keys or ordering keys.

_ **UniqueKey** a list of attributes on which instances of the base class may be keyed.

subtype_of: Key

emoji:

label: Note

content: order unimportant for Unique Keys.

Attributes

_ **Attribute Section**

a group of attributes for a class that merit a shared explanation.

subtype_of: Component.
based_on: Class

- **isOptional** whether the attributes in this section, taken together, are optional. *Boolean*

  content: If the Attribute Section is required,then each Attribute within the sectional is optional ot required, depending on how it is marked. + + But if the Arrribute Section is optional each attribute in the section is only required if any attribute in the section is ptresent.

__ **Markdown Support**

- **mdPrefix** **[String](#string*

  default: **AS_ENTERED:** "_ "

- **mdTopLine** **[String](#string*

_ **AttributeA property or characteristic of a class**

plural: Attributes
subtype_of: Component
based_on: AttributeSection

- **name** *Lower Camel*

  overrides: **CLASS_NAME:** CamelName
  ATTRIBUTE_NAME:

- **dataType** The kind of object to which the attribute refers. _ *DataType*

  content: But,

- **List of Editions**

- **Set of Edition**

- **... and more complicated cases.**

  emoji:

  label: See

  content: the section below on Data Type Specifiers.

__ **Cardinalities.**

- **isOptional** Indicates whether the attribute must have a value for every instance of the class _ ***Boolean***

  default: **AS_ENTERED:** *** False

- **cardinality** The cardinality of the relationship represented by the attribute _ ***CardinalityCode***

  default: **AS_ENTERED:** *** For a singular attribute, the default cardinality is N:1. If the attribute is 1:1, it must be stated explicitly.For a collective attribute, the default is 1:N. If the attribute is N:M, it must be stated explicitly.

  CONTENT: ***DSL***: the cardinality of an attribute, if stated explicitly, should be placed just before the class name in the parenthetical data type

specification after the one-liner.

emoji:

label: For example

content:

- **author** *1:1 Author*

- **books** *optional N:M Set of Books*

emoji:

label: Note

content: how this works with optionality

__ **Inverse Attributes**

- **isInvertible** *Boolean*

derivation: **AS_ENTERED:** true if the data type is a class or a simple collection of members of a class.

- **inverseClass** the class which contains, or would contain the inverse attribute *optional Class*

derivation: **AS_ENTERED:** from the data type. Null unless arrribute is invertible.

- **inverseAttribute** *optional Attribute*

- **inverseIsOptional** *optional Attribute*

_ **Formulas**

- **default** The rule or formula for calculating the value, if no value is suppliedNow running to a second *Derivation, optional*

line with the parenthentical on yet
a third line -

emoji:

label: Note

content: even when an Attribute has a default derivation, thereâ€™s no guarantee that every instance will have an assigned value. Example needed.And let's see if the note can span extra lines, too

content: Yes, it handled extra lines. Let's see about additional paras for an annotation

content: Last paragraph here

- **derivation** For derivedattributes, the rule or formula for calculating the value _ ***Derivation, optional***

emoji:

label: Issue

content: on insert vs on access?

- **constraints** Any validation rules specific to this attribute _ ***ListOf Constraints***

emoji:

label: Note

content: from Class.constraints

__ **Override Tracking**

- **Overrides**

_ **ValueType: DerivationA rule or formula for deriving the value of an attribute**

plural: Derivations

- **statement** An English language statement of the derivation rule _ ***RichText***

- **expression** The formal expression of the *CodeExpression* derivation in a programming language _

_ **ValueType: ConstraintA rule, condition, or validation that must be satisfied by the model**

plural: Constraints

subtype_of: **Component**

- **statement** An English language statement of the *RichText* constraint _

- **expression** The formal expression of *e.g., OCL* the constraint in a *_(CodeExpression* programming language

- **severity** *Code*

- **Warning** - nothing fatal; just a caution

- **Error** - serious. Fix now

- **Message** *Template*

_ **Class Constraint**

subtype_of: **Constraint**

based_on: **Class.**

_ **Attribute Constraint**

subtype_of: **Constraint**

based_on: **Attribute**

_ **CodeExpression**

- **Language** the programming language *Code*

- OCL: Object Constraint Language

- Java: Java

- **Expression** *String*

Methods

_ **MethodA behavior or operation associated with a class**

plural: Methods

subtype_of: **Component**

- **parameters** The input parameters of the method _ *ListOf Parameters*

- **returnType** The data type of the value returned by the method _ *DataType*

_ **ParameterAn input to a method**

plural: Parameters

subtype_of: **Component**

- **type** The data type of the parameter _ *DataType*

- **cardinality** The cardinality of the parameter *e.g., optional, required*

Data Types

content: *ValueType*:**Data Type**

_ **Simple Data TypeSubtpeOf: DataType**

- **coreClass** *Class*

_ **Complex Data Type**

- **aggregation** *Aggregating Operator*

- **aggregatedTypes** *List of DataTypes*

_ **Aggregating Operator**

- **Name** *Code*
- **SetOf**
- **ListOf**
- **Mapping**
- **arity** *Integer*
- **spelling** *Template*

Low level Data Types

content: insert Camel Case.md

_ **ValueType: CamelName**

content: A short string without punctuation or spaces, suitable for names, labels, or identifiers and presented in camel case.

subtype_of: **String**

- **value: the string** *String*

as_entered: Must follow the camel case naming convention and not be empty.

emoji:

label: Example

content: "firstName", "orderDate", "customerID"

content: > ðŸ“ ***ModelingNote***: Putting the non-empty constraint on the CamelName value type is effective because it automatically applies to all attributes that use CamelName as their type. This ensures consistency and avoids the need to define the constraint separately for each attribute.

emoji:

label: ModelingNote

content: * *CamelName* is presented here, just after its first usage by another class (Component), to provide context and understanding before it is used further in the model.

_ **UpperCamel** a CamelName that begins with a capital letter

subtype_of: CamelName

where: content begins with an upper case letter.

emoji:

label: Example

content: _ "Customer", "ProductCategory", "PaymentMethod"

_ **LowerCamel** a CamelName that begins with a lower case letter

subtype_of: CamelName

where: content begins with a lower case letter.

emoji:

label: Example

content: "firstName", "orderTotal", "shippingAddress"

_ **Qualified Camel** an expression consisting of Camel Names separated by periods

subtype_of: String

as_entered: content consists of CamelNames, separated by periods. Each of the camel names must be Upper Camel except, possibly, the first.

_ **RichText. A string with markup for block level formatting.**

subtype_of: String

- **value** the string content *string*
- **format** the rich text coding language used *Code*
- **HTML**

- **MarkDown**

_ **RichLine** String with markup for line level formatting.

subtype_of: [RichText]

- **value** the string content *string*

as_entered: must not containa line break or new line character

_ **PrimitiveType**

subtype_of: [ValueTypeA basic built-in data type]

emoji:

label: Values

content: **String**

content: ===

AppendicesInsert Insert Overrides.md - insert LDM Intro.md - **More Sidebars.md** Insert OCL.md - Insert Camel Case.md

Annotation Types Used

content: These are the recognized Annotation Types for the LDM model.

content: And this is how you register the AnnotationTyped for a model. By including this sort of array in the DSL document for the model.

content:

```typescriptinterface
AnnotationType {
 - label: string;
 - emoji: string;
 - emojiName: string;
 - emojiUnicode: string;
 - purpose: string;
 - }
 - // LINK:
LiterateDataModel.annotationTypes
 - const annotationTypes:
```

```
AnnotationType[] = [
 - {
 - label: "Error",
 - emoji: "",
 - emojiName: "cross_mark",
 - emojiUnicode: "U+274C",
 - purpose: "Indicates a critical
error or failure in the model."
 - },
 - {
 - label: "Warning",
 - emoji: "",
 - emojiName: "warning",
 - emojiUnicode: "U+26A0",
 - purpose: "Indicates a potential
issue or warning in the model."
 - },
 - {
 - label: "Note",
 - emoji: "",
 - emojiName: "blue_book",
 - emojiUnicode: "U+1F4D8",
 - purpose: "Provides additional
context, explanations, or
clarifications for the annotated
element."
 - },
 - {
 - label: "Issue",
 - emoji: "",
 - emojiName: "warning",
 - emojiUnicode: "U+26A0",
 - purpose: "Highlights a potential
issue or error that needs to be
addressed or resolved."
 - },
 - {
 - label: "Question",
 - emoji: "",
 - emojiName: "question",
```

```
 - emojiUnicode: "U+2753",
 - purpose: "Raises a question or
seeks further clarification about
the annotated element."
 - },
 - {
 - label: "Suggestion",
 - emoji: "",
 - emojiName: "bulb",
 - emojiUnicode: "U+1F4A1",
 - purpose: "Provides a suggestion
or recommendation for improving the
model or the annotated element."
 - },
 - {
 - label: "Info",
 - emoji: "",
 - emojiName: "information_source",
 - emojiUnicode: "U+2139",
 - purpose: "Offers relevant
information, facts, or details
about the annotated element."
 - },
 - {
 - label: "Todo",
 - emoji: "",
 - emojiName: "pushpin",
 - emojiUnicode: "U+1F4CC",
 - purpose: "Indicates a pending
task, action item, or future work
related to the annotated element."
 - },
 - {
 - label: "Reference",
 - emoji: "",
 - emojiName:
"globe_with_meridians",
 - emojiUnicode: "U+1F310",
 - purpose: "Provides a reference
or link to an external resource or
```

```
documentation."
 - },
 - {
 - label: "See",
 - emoji: "",
 - emojiName: "mag",
 - emojiUnicode: "U+1F50D",
 - purpose: "Indicates a cross-
reference to another relevant
element within the model."
 - }
 - ];
 - ```
```

content: ===

AppendicesInsert Insert Overrides.md - insert LDM Intro.md - **More Sidebars.md** Insert OCL.md - Insert Camel Case.md