The Legal-RDF Ontology. A Generic Model for Legal Documents

John McClure
Legal-RDF.org / Hypergrove Engineering
jmcclure@hypergrove.com

Abstract. Legal-RDF.org[1] publishes a practical ontology that models both the layout and content of a document and metadata about the document; these have been built using data models implicit within the HTML, XSL, and Dublin Core dialects. Directed Acyclic Graphs (DAGs) form the foundation of all models within the ontology, that is, *DAGNode* and *DAGModel* are the base classes for all other ontology classes, which include a restatement of RDF and OWL classes and properties as well as basic Kellog parts-of-speech. The ontology also represents an explicit semantic model used during its classifications: concrete classes are categorized as some element of a dramatic production, that is, as a subclass of *Actor*, *Role*, *Scene*, *Prop*, *Theme*, or *Drama*; this can be helpful during analyses of semantic perspective and context associated with resource definitions and attribute values. The Legal-RDF ontology distinguishes between predicate verbs and predicate nouns in its models of a Statement to yield an intuitively appealing vocabulary that segregates attributes as past, present, future, or conditional, information. To facilitate development of generic tools, all data and object properties defined in the ontology's models are categorized as a subproperty of one of the 15 Dublin Core properties; provenance data, with emphasis on an asOf timestamp, may be recorded for any attribute of a resource. Legal-RDF's numeric properties derive from the ISO Systeme Internationale measurement systems; algebraic properties derive from XML Schema datatypes; language and currency designations are based upon relevant ISO standards; and time-zone designations are based on a review of local and regional standards (with some modifications necessary to eliminate collisions between the names of these properties and ISO standards). In addition to classes that represent quantities, classes are included that represent qualities that may be used to subtype or otherwise characterize instances.

Keywords: Aspect-oriented programming, Dublin Core, Kellog Grammar.

1. Status of the Legal-RDF Ontology

Version 2 of the Legal-RDF ontology – which this paper describes – is being documented in a Wiki[2] hosted by LexML.org[3] to encourage the participation of an interested community during its development. The first version of the ontology, at the Legal-RDF.org website, is being

[1] `http://www.hypergrove.com/legalrdf.org/index.html`

[2] The wiki is located at `http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF\_Ontologies`

[3] http://www.lexml.org/

improved in Version 2 with the adoption of a better class- and property-naming guideline; with the refactoring of base/derived classes; and with the definition of more complete data models.

2. Naming Conventions

Facilitating the construction of *dotted-names* is the primary objective of Legal-RDF class and property naming guidelines. A dotted-name defines a path in a Directed Acyclic Graph, e.g., *Person.FullName.FirstName*, intentionally aligned with the ECMA-242 standard for attribute value references. In effect, the Legal-RDF ontology aims to define an object model that is reasonable in the context of software written in an ECMA language, e.g., Javascript, C#, and Eiffel.

Text properties are in lower-case, e.g., *Person.FullName.FirstName.eng* is a reference to a string of English text while *Person.FullName.FirstName* references an object (i.e., a resource) for which the *eng* text property may be present.

RDF triples are accommodated by a defaulting mechanism. When no predicate is specified, the *has* predicate is implied, e.g., the dotted-name above transforms to *Person.has.FullName.has.FirstName.eng*, allowing the use of other predicate verbs such as *Person.willHave.FullName* to describe, perhaps, a bride.

A consequence of this approach is that Legal-RDF separately defines predicate verbs and predicate nouns, specifically eschewing the RDF community practice that concatenates these as single property names, e.g., "hasName". This yields a naming system that is historically more familiar to the software industry, while maintenance economies are had as new predicates are defined over time. [4]

All Legal-RDF classes are named by at least two words so that, when the class is the range of an object property, a single word can be used for the property name.

3. The CoreResource[5]Class

All classes in the ontology derive from the *CoreResource* class whose function is to allow Dublin Core attributes to be associated with any resource. This class demonstrates how the Legal-RDF ontology incorporates the principles of aspect-oriented programming. The ISO Dublin

[4] An "RDF quint" composed of subject, predicate verb, predicate noun, object, and node identifier.

[5] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:CoreResource

Core properties are not properties of the *CoreResource* class; instead the Dublin Core properties are inherited from superclasses representing qualities had by instances of *CoreResource*. To the maximum extent possible, all properties in the Legal-RDF ontology are bundled as quality classes, enabling (a) creation of aspect-oriented applications (b) adoption of other namespaces (c) clearer simpler class hierarchies and (d) comparative quality analyses.

Table I. *CoreResource* Inherited Properties

CoreResource **Superclass**[6]	**DublinCore Property**	*CoreResource* **Superclass**	**DublinCore Property**
CategorizableThing	subject	IdentifiableThing	identifier
ClassifiableThing	type	ManagableThing	rights
CreatableThing	creator	NamableThing	title
DerivableThing	source	PublishableThing	publisher
DescribableThing	description	RelatableThing	relation
EnhanceableThing	contributor	SchedulableThing	date
ExpressibleThing	language	ScopableThing	coverage
		StylableThing	format

The inherited properties (e.g., subject) listed in Table 1 are text-properties. Paired with each superclass, e.g., *ClassifiableThing*, is a subclass representing the **state** of a resource with respect to the quality, e.g., *ClassifiedThing* is a proper subclass of *ClassifiableThing* in that every thing that is deemed 'classified' in some way is, by absolute semantic necessity, a 'classifiable' thing. The *ClassifiedThing* class includes a *Type* property whose range is *ClassNode* (read: *owl:Class*). An instance of a *CoreResource* is therefore not a *ClassifiedThing* until the text within a *type* attribute has been correlated with a class defined by the ontology.
The *CoreResource* class has no object properties and only two text properties: *asOf,* a timestamp to record when a resource was last updated; and *rdf*, a URI for retrieving an RDF representation of the resource.

4. The *DAGNode*[7] and *DAGModel*[8] Classes

[6] These classes are subclasses of the *CapabilityFacet* class, and are each paired with a subclass of the *StateFacet* class – both are subclasses of a *FacetNode* class, a subclass of the *DAGNode* class, which is itself a subclass of the *CoreResource* class. The circularity of this hierarchy is an open issue.
[7] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:DAGNode
[8] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:DAGModel

The *rdf:Description* class is the primary superclass for the *DAGModel* class so as to clearly delineate its role as a representation of a Directed Acyclic Graph (DAG). DAGs are composed of nodes and arcs; arcs are classified as either terminating with a node or with a literal text string.

DAGModel and *DAGNode* derive from *CoreResource* and from classes that correspond to the Representational State Transfer (REST) protocol:

(a) the *DeletableThing* class, with its *DeletedThing* subclass, correspond to a potential or actual Delete operation;

(b) the *RecordableThing/RecordedThing* classes correspond to a Put;

(c) the *RetrievableThing/RetrievedThing* classes correspond to a Get;

(d) the *UpdatableThing/UpdatedThing* classes correspond to an Update.

All provenance data about creation, retrievals, updates, and deletions of a resource or resource attribute are captured by instances of these classes.
DAGModel subclasses are established to correspond with the types of Unified Modeling Language (UML) diagrams. An 'ontology' for example corresponds to a *ClassModel*. Second, subclasses exist for document types; page layouts; and for specifying 'one-off' resource instance models.

Table II. *DAGModel* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Arc	StatementNode	Coverage	CoreResource	ClassModel
LiteralArc	StatementNode	Arc	rdf:Description	DocumentModel
Node	DAGNode	Coverage	DeletableThing	EventModel
ObjectArc	StatementNode	Arc	RecordableThing	InstanceModel
			RetrievableThing	PageModel
			UpdatableThing	ProcessModel

DAGNode subclasses correspond to classes defined by the RDF, RDF Schema, and OWL specifications. Beyond these, two additional subclasses are defined, *ContextNode* and *FacetNode*.

Table III. *DAGNode* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Arc	StatementNode	Coverage	CoreResource	ClassNode
Model	DAGModel	Source	CountableThing	CollectionNode
Slot	FacetNode	Description	DeletableThing	ContextNode
Template	InstanceModel	Format	RecordableThing	FacetNode
			RetrievableThing	LiteralNode
			UpdatableThing	PropertyNode
				StatementNode

5. The *LiteralNode*[9] Class

The ontology defines the *LiteralNode* class to represent words, sounds, figures, images, or video clips that can be rendered for presentation. Properties of the *LiteralNode* class allow a concept represented by an instance to be simultaneously expressed in words, sounds, figures, images, and or video. The *Content* super-property is defined for the *ExpressedThing* class, itself a subproperty of the *ExpressedThing* class' *Language* property.

Table IV. *LiteralNode* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Audio	AudioNode	Content	DAGNode	AudioNode
Graphic	GraphicModel	Content	AnalyzableThing	GraphicModel
Image	ImageNode	Content	ExpressibleThing	ImageNode
Text	TextNode	Content	HidableThing	TextNode
Video	VideoNode	Content		VideoNode

All document text content and nearly all text properties associated with any resource are represented using the *TextNode* subclass. The *TextNode* class has two groups of subclasses: (a) ones relating to functional types of document text, e.g., strings of text, text tokens, symbolic text, etc. and (b) the union of classes that represent upper-, lower-, and mixed-case text.

NumericText, a subclass of *TextNode*, uses the subclass, *RealNumber*, to represent all real numbers found in documents.

The *RealNumber* class defines the *float* text property, which corresponds to the *float* attribute defined by XML Schema Datatypes. Legal-RDF's *float* property is a subproperty of the *value* text property

[9] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:LiteralNode

Table V. *TextNode* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Language	LanguageFacet	Type	rdf:Literal	NumericText
Length	IntegerNumber	Description	LiteralNode	SemanticText
text	xmls:string	(none)	LinkableThing	SymbolicText
			PaddableThing	TextBlock
			TintableThing	TextString
			TypesettableThing	TextToken

Table VI. *NumericText* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
(none)			TextNode	ComplexNumber
			ComparableThing	ImaginaryNumber
			ConvertableThing	RealNumber
			EstimableThing	NonNegativeNumber
			QuantifiableThing	NonPositiveNumber
			RoundableThing	StatisticalNumber
			ScalableThing	

for the *QuantifiedThing* class. In other words, when a numeric value is provided as an attribute, the attribute is then deemed to have entered the 'quantified' state.

Table VII. Systeme Internationale Classes

Subclass	***QuantityFacet* Subclass' Subclasses**
Angle-Measure	ArcDegreeQuantity, ArcMinuteQuantity, ArcSecondQuantity, RadianQuantity, SteradianQuantity
Area-Measure	AcreQuantity, AreQuantity, CentiareQuantity, ColumnInchQuantity, CommercialAcreQunatity, HectareQuantity, SquareCentimeterQuantity, SquareQuantity, SquareFootQuantity, SquareInchQuantity, SquareYardQuantity, SquareKilometerQuantity, SquareMeterQuantity, SquareMileQuantity, SquareMillimeterQuantity
Capacity-Measure	BarrelQuantity, CentiliterQuantity, CubicCentimeter, LiterQuantity, MilliliterQuantity, CubicKilometer, CubicMeterQuantity, CubicMillimeter
Density-Measure	RadQuantity, TeslaQuantity, WeberQuantity
Distance-Measure	AngstromQuantity, CaliberQuantity, CentimeterQuantity, DecimeterQuantity, EmQuantity, FootQuantity, FortyFootEquivalentQuantity, FurlongQuantity, GaugeQuantity, InchQuantity, KilometerQuantity, MeterQuantity, MileQuantity, MillimeterQuantity, NauticalMileQuantity, PointQuantity
Dry-Measure	BaleQuantity, BoardFootQuantity, BundleQuantity, BushelQuantity, CartonQuantity, CordQuantity, CubicFootQuantity, CubicInchQuantity, CubicMileQuantity, CubicYardQuantity, DozenQuantity, DryQuartQuntity
Electrical-Measure	AmpereQuantity, CoulombQuantity, FaradQuantity, GigawattHourQuantity, GigawattQuantity, JouleQuantity, MegawattHourQuantity, MegawattQuantity, WattQuantity, MilliwattHourQuantity, MilliwattQuantity, OhmQuantity, SiemansQuantity, TerawattQuantity, VoltQuantity
Energy-Measure	BritishThermalUnitQuantity, CalorieQuantity, GrayQuantity, HorsepowerQuantity, KilopascalQuantity, NewtonQuantity, PoundsPerSquareFootQuantity, PoundsPerSquareInchQuantity
Frequency-Measure	CyclesPerMinuteQuantity, CyclesPerSecondQuantity, GigahertzQuantity, HertzQuantity, KilohertzQuantity, MegahertzQuantity
Light-Measure	CandelaQuantity, LumenQuantity
Medical-Measure	InternationalUnitQuantity, KatalQuantity, SievertQuantity
Pressure-Measure	BarQuantity, BecquerelQuantity, DecibarQuantity, DyneQuantity, HectopascalQuantity, KilopascalQuantity, PascalQuantity
Sound-Measure	DecibelQuantity, SabinQuantity
Speed-Measure	FeetPerMinuteQuantity, FeetPerSecondQuantity, KnotQuantity, KilometersPerHourQuantity, MetersPerSecondQuantity, MilesPerHourQty
Temperature-Measure	CelsiusQuantity, FahrenheitQuantity, KelvinQuantity, ThermQuantity
Time-Measure	DayQuantity, DecadeQuantity, HourQuantity, MinuteQuantity, MonthQty, QuarterQuantity, SecondQuantity, WeekQuantity, YearQuantity
Velocity-Measure	CubicCentimetersPerSecondQuantity, CubicFeetPerMinuteQuantity, CubicMetersPerHourQuantity, CubicMetersPerSecondQuantity, GallonsPerMinuteQuantity
Volume-Measure	AcreFootQuantity, CupQuantity, FluidOunceQuantity, GallonQuantity, ImperialGallonQuantity, PintQuantity, QuartQuantity
Weight-Measure	AssayTonQuantity, CaratQuantity, CentigramQuantity, GrainQuantity, GramQuantity, KilogramQuantity, MilligramQuantity, MoleQuantity, OunceQuantity, PoundQuantity, PoundFootQuantity, StoneQuantity, TonQuantity, TroyOunceQuantity, TroyPoundQuantity, TonneQuantity

The *RealNumber* class illustrates another feature of the ontology. It is a superclass of the *QuantityFacet* class, a subclass of the *FacetNode* class (see Table XVI). *QuantityFacet* defines subclasses that correspond to all measures standardized by the Systeme Internationale (SI). Each has a text property – whose name matches SI's standard abbreviation for the measurement – that is a subproperty of the *QuantifiedThing* class' *value* property, e.g., *m2* is the text property defined for the *SquareMeterQuantity* class, where 'm2' is the SI name for 'square meter' measurements.

Finally in the area of numerics, classes exist for each currency recognized by the ISO. For instance, the *UnitedStatesDollarAmount* class derives

Table VIII. *SemanticText* Class Relations

Properties	Range	Category	Superclasses	Subclasses
Abbreviation	AbbreviationToken	Title	TextNode	TextPhrase
Acronym	AcronymToken	Title		TextWord

from the *CurrencyAmount* class, which derives from *DecimalAmount*, which derives from *ProperFraction*, which derives from *FractionalNumber*, deriving from the *RealNumber* class identified in Table VI as a subclass of *NumericText*.

CurrencyAmount is the superclass for *EconomicAmount* and *EconomicValue*. These classes measure certain currency flows (*CapitalAmount, ChargeAmount, DeductionAmount, DiscountAmount, DueAmount, Expense-Amount, IncomeAmount, LiabilityAmount, NetAmount, NetWorthAmount, PriceAmount, ProfitAmount, RevenueAmount, WealthAmount*). A number of these subclasses are decomposed by economic factors of production, e.g., C*hargeAmount* has the subclasses *CapitalCharge, LaborCharge, Material-Charge, ProductCharge,* and *ServiceCharge*.

The *SemanticText* class is notable because it unions more than 140 classes that represent each of the languages defined by ISO-639, replicating functionality of the *xml:lang* attribute. For example, the *EnglishText* class defines the *eng* property whose super-property is the *text* property defined for the *TextNode* class.

The *TextPhrase* class segues to Legal-RDF's linguistic model, as it is the superclass for the *TextClause, AdjectivePhrase, AdverbPhrase, Noun-Phrase, Verb-Phrase, PrepositionPhrase,* and *InterjectionPhrase* classes. *TextClause* is the superclass for the *TextSentence* class, which is the superclass for two classes, *CompoundSentence* and *ComplexSentence*.

Table IX. *TextClause* Class Relations

Properties	Range	Category	Superclasses	Subclasses
Adjective	AdjectivePhrase	Content	TextPhrase	TextSentence
Nominative	NounPhrase	Content		
Object	NounPhrase	Content		
Predicate	TextClause	Content		
Subject	NounPhrase	Content		
Verb	VerbPhrase	Content		
DirectObject	NounPhrase	Object		
IndirectObject	NounPhrase	Object		
Punctuation	PunctuationMark	Format		

The *TextBlock* class represents a *TextNode* that is visually distinct from a simple string of text characters, laid out in a rectangular fashion, and is the gateway to block-elements defined by the XHTML 2.0 and XSL dialects.

Table X. *TextBlock* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Body	TextBody	Source	TextNode	TextObject
Statement	StatementNode	Description	ParsableThing	TextPage
Watermark	ImageNode	Format	RectangularThing	

In the model above, a *TextBlock* is part of a *TextBody*; can have an *ImageNode* specified as a *Watermark*; and may have multiple *StatementNode* instances to describe the contents of the *TextBlock*. Two subclasses are defined, *TextObject* and *TextPage,* whose qualities and properties differ.

Table XI. *TextObject* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Page	TextPage	Source	TextBlock	TextBody
			AnnotatableThing	TextColumn
			ApprovableThing	TextDivision
			CitableThing	TextHeading
			DraftableThing	TextImage
			IllustratableThing	TextIndex
			PositionableThing	TextLine
			PrintableThing	TextList
			ReviewableThing	TextParagraph
			TranslatableThing	TextQuote
			VersionableThing	TextRow
			ViewableThing	TextSection
				TextTable

The class model in Table XI allows a *TextObject* instance to be located on zero or more pages, and its subclasses reflect its coverage of XHTML 2.0 elements. Its qualities indicate typical document actions are permitted, that is, *TextObject* instances can be annotated, approved, cited, drafted, illustrated, positioned, printed, reviewed, translated, versioned, or viewed.

Properties in the *TextPage* class model (Table XII) indicate six layout areas can be formatted. Around the *BodyArea* may be arrayed a *BannerArea*, *FooterArea*, *HeaderArea*, *SidebarArea*, and *SignatureArea*.

Each area has an associated reference to the text, image, sound, figure, or video content to be placed into the area. This "page model" can be customized for a particular document type by a reference to a *PageModel* (a subclass of the *DAGModel* class), which has names and positioning of custom areas that can be displayed in a user. Finally, note that the pagination-specific Cascading Stylesheet (CSS) *size* text attribute can be specified for a *TextPage*, as part of the CSS-2 support provided.

Table XII. *TextPage* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Banner	LiteralNode	Content	TextBlock	BlankPage
BannerArea	RectangularThing	LayoutArea	FoliatableThing	TitlePage
Body	LiteralNode	Content	ViewableThing	
BodyArea	RectangularThing	LayoutArea		
DotPage	TextPage	Relation		
Footer	LiteralNode	Content		
FooterArea	RectangularThing	LayoutArea		
Header	LiteralNode	Content		
HeaderArea	RectangularThing	LayoutArea		
LayoutArea	PositionableThing	Format		
MappingModel	PageModel	Format		
Sheet	PaperSheet	Source		
Sidebar	LiteralNode	Content		
SidebarArea	RectangularThing	LayoutArea		
Signature	LiteralNode	Content		
SignatureArea	RectangularThing	LayoutArea		
NextPage	TextPage	Relation		
PreviousPage	TextPage	Relation		
size	xmls:string	style		

The *TextBody* class is equivalent to HTML's *body* element and allows one to specify the default header, footer, and banner content to appear on pages in the document when it is formatted. The model has properties for front- and rear-matter in the document, and for other functional document parts (colophon, notes, bibliography, etc) not allocated to a quality class.

The qualities associated with *TextBody* instances enable specification of typical functional document parts, including its attachments; its tables of contents, of figures, of tables, and of authorities; its internal divisions and sections; its cover pages; and its introductory and conclusion material.

Table XIII. *TextBody* Class Relations

Properties	Range	Category	Superclasses	Subclasses
Banner	LiteralNode	Relation	TextObject	(none)
Bibliography	TextIndex	Source	OntologicalThing	
Colophon	LiteralNode	Description	AppendableThing	
EndNote	LiteralNode	RearMatter	AttachableThing	
Epilogue	LiteralNode	RearMatter	ConcludableThing	
Footer	LiteralNode	Relation	IndexableThing	
FrontMatter	LiteralNode	Content	IntroducibleThing	
Header	LiteralNode	Relation	PaginatableThing	
Paragraph	TextParagraph	Content	PrintableThing	
RearMatter	LiteralNode	Content	PrependableThing	
SubTitle	TextHeading	SecondaryTitle	StaplableThing	
			SubDivisableThing	
			SubSectionableThing	

6. The *FacetNode*[10] Class

Document pagination can demonstrate how quality classes play a key role in the specification (and validation) of an instance model. To begin, a *TextBody* is a *PaginatableThing*, that is, its content can be formatted across one or more pages. When formatting occurs, the *TextBody* instance is assigned these *Page* attributes, each referencing a *TextPage* instance; only then does the *TextBody* instance enter the 'state' of being a *PaginatedThing*. A *TextBody* thus has the capacity to be paginated, as is so indicated by its superclass *PaginatableThing*; it is only after pagination that it is explicitly or deducibly a *PaginatedThing.*

Table XIV. *PaginatableThing* Class Relations

Properties	Range	Category	Superclasses	Subclasses
PaginationPlan	PredictiveStatement	Plan	CapabilityFacet	PaginatedThing
PaginationPolicy	RequirementStatement	Policy		

The classes above derive from *FacetNode*, a subclass of *DAGNode* (see Table III). In the Legal-RDF ontology, a facet is "an instance of an attribute value; a named value or relationship". Five types of resource facet are identified:

(a) its capabilities, e.g., "the resource can be deleted"

[10] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:FacetNode

Table XV. *PaginatedThing* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Pagination	PaginationEvent	EntryEvent	PaginatableThing	(none)
Page	TextPage	Format	StateFacet	

(b) its existence, e.g., "the resource is an American resource"

(c) its qualities, e.g., "the resource is expressed in English"

(d) its numeric quantities, e.g., "the resource has five attributes" and

(e) its states of existence, e.g., "the resource was validated".

Table XVI. *FacetNode* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Collection	CollectionNode	Subject	DAGNode	CapabilityFacet
Literal	LiteralNode	Language		ExistentialFacet
Object	CoreResource	Relation		QualityFacet
Property	PropertyNode	Title		QuantityFacet
Verb	rdf:Property	Coverage		StateFacet

7. The *StatementNode*[11] Class

The *StatementNode* class plays a central role in the Legal-RDF ontology in two ways. The class defines components of the proposed "RDF quint", extending the now-classic "RDF quad" with explicit specification of the predicate-verb for an instance. The class additionally defines properties for the context of and the source for the statement. The Legal-RDF process model envisions that: (a) a document when drafted, yields 'content'; (b) content when annotated, yields 'identities'; (c) content when parsed, yields 'sentences' and (d) sentences when normalized using identities, yields 'statements'. This process model implies that document content contains both sentences and statements, the latter being a formally structured characterization of the former.

A role of the *StatementNode* is to package the subclasses that define predicates verbs appropriate to the particular type of statement. The

[11] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:StatementNode

Table XVII. *StatementNode* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses, cont'd**
Context	ContextNode	Description	DAGNode	FactualStatement
Model	DAGModel	Source	rdf:Statement	FictionalStatement
Object	CoreResource	Coverage		HistoricalStatement
Predicate	PropertyNode	Coverage	**Subclasses**	HypotheticalStatement
TextSource	TextBlock	Source	AcknowledgingStatement	ParentheticalStatement
Verb	rdf:Property	Coverage	CautionaryStatement	PermissiveStatement
			ConclusiveStatement	PredictiveStatement
			ConditionalStatement	ProhibitiveStatement
			ConsequentialStatement	RequestStatement
			DefinitionalStatement	RequirementStatement
				ResponseStatement

ConditionalStatement defines three properties – *If, Then,* and *Else* – whose ranges are *StatementNode*. Similarly, the *ConsequentialStatement* defines two properties – *When* and *Then* – having the same range.

Table XVIII. Legal-RDF Predicate Verbs[12]

***StatementNode*Subclass**	**Predicate Verbs**
CautionaryStatement	mayNotBeA, mayNotHaveBeenA, mayNotHave, mayNotHaveHad
DefinitionalStatement	isA, isNotA, wasA, wasNotA, willBeA, willNotBeA
FactualStatement	had, hadNot, has, hasNot, willHave, willHaveNot
ProhibitiveStatement	shallNotBeA, shallNotHave, shallNotHaveHad
PermissiveStatement	canBeA, canBeNotA, canHave, canHaveNot, canHaveHad, canHaveHadNot
RequirementStatement	mustBeA, mustHave, mustHaveBeenA, mustHaveNotBeenA, mustHaveHad, mustHaveNot, mustHaveHadNot, mustNotBeA

8. The *ContextNode*[13]Class

The Legal-RDF ontology adopts a 'thematic' perspective in its organization of OWL classes for people, places, and things; the objective is to establish a strong guideline useful during the classification process. We observe a reality, as does an audience. These **concrete ontology classes** are categorized as one does for the elements of a play: we distinguish between the actors and their roles, between scenes and their props,

[12] This table needs refinement.

[13] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:ContextNode

and between the themes communicated by its dramas. An element of a play establishes a context for the interplay of other elements. Accordingly, the *ContextNode* class, a subclass of the *DAGNode* class, has six subclasses: *ActorContext, DramaContext, Prop-Context, RoleContext, SceneContext,* and *Theme-Context.* A perspective that includes the time and venue of the play, its production and other aspects, is already implicit in provenance information relating to the resources described and the attribute values provided.

9. The *GenericDocument*[14] and *GenericLegalDocument*[15] Classes

Table XIX shows two sets of subclasses for the *GenericDocument* class: formal subclasses (including quality classes) and second, union subclasses comprised of documents associated with economic industrial sectors[16]. The set of formal subclasses is distinguished by the type and size of the physical sheet of paper used to print and bind the contents of the document.

The *GenericLegalDocument* class represents documents that historically have been printed on legal-type of paper. It has just two properties: *Clause* and *Rider*, both of which have a range of *GenericClause.* The *GenericClause* class provides subclasses for standard types of clauses, e.g., a *DamagesClause* that could appear within a lease contract. The *Generic-Clause* class derives from the *GenericRule* class, which derives from the *LiteralNode* class.

[The *GenericRule* class also has subclasses *GenericLaw, GenericByLaw,* and *GenericRegulation.* The *GenericLaw* class subdivides to *GenericCivilLaw, GenericCriminalLaw, GenericMaritimeLaw*, and *GenericMilitaryLaw.* This class also features a union of *GenericCaseLaw, GenericCommonLaw, GenericStatutoryLaw*, and *GenericTreatyLaw* where classes exist for national, state, local and international entities to classify instances of their laws. Within the subclass for criminal law, the LegalRDF ontology has subclasses for criminal assistance and criminal conspiracy laws, while it unions commercial, economic, interpersonal, offensive, and violent laws; these have all been derived by analysis of the US justice system database structure. Each type of law is then divided into

[14] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:GenericDocument

[15] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:GenericLegalDocument

[16] These sectors coincide with those defined by the North American Industrial Classification System

subclasses for an act of violation, an activity of violation, an attempt at violation, and a threat of violation, of the law.]

Table XIX. *GenericDocument* Class Relations[17]

Superclasses	**Subclasses**	**Unions**
DocumentModel	GenericBook	AdministrativeDocument
ApprovableThing	GenericBooklet	AgricultureDocument
ArchivableThing	GenericBlueprint	CommunityServiceDocument
FilableThing	GenericCalendar	ConstructionDocument
DistributableThing	GenericCard	EducationDocument
ReviewableThing	GenericCertificate	EntertainmentDocument
	GenericFoil	FinancialDocument
	GenericLabel	HealthCareDocument
	GenericLedger	HospitalityDocument
	GenericLegalDocument	InformationDocument
	GenericMagazine	ManagementServiceDocument
	GenericMap	ManufacturingDocument
	GenericNewspaper	MiningIndustryDocument
	GenericPoster	ProfessionalServiceDocument
	GenericSlip	PublicAdministrationDocument
	GenericStationery	RealtyAndLeasingDocument
		RepairServiceDocument
		RetailTradeDocument
		TransportationWarehousingDocument
		UtilityIndustryDocument
		WholesaleTradeDocument

[17] This class has just one property, not shown: *Body*, a *TextBody*, categorized as *Content*.

Table XX. *GenericLegalDocument* Selected Subclass Hierarchy[18]

Subclasses	**Subclass' Subclasses**
GenericAffidavit	GenericAffidavitOfDefense, GenericAffidavitOfInquiry, GenericAffidavitOfMerits, GenericAffidavitOfNotice, GenericAffidavitOfService, GenericAffidavitOfTitle, GenericAffidavitToHoldBail
GenericCharter	GenericBankCharter, GenericCityCharter, GenericCorporationCharter
GenericLegislation	GenericLegislativeBill, GenericLegislativeLaw, GenericLegislativeResolution
GenericWrit	LegalWritOfArraignment, LegalWritOfAttachment, LegalWritOfCertiorari, LegalWritOfDecree, LegalWritOfElection, LegalWritOfDefaultJudgment, LegalWritOfDeficiencyJudgment, LegalWritOfDetinue, LegalWritOfError, LegalWritOfExecution, LegalWritOfFieriFacias, LegalWritOfDecreeOfForeclosure, LegalWritOfHabeusCorpus, LegalWritOfIndictment, LegalWritOfInjunction, LegalWritOfMandamus, LegalWritOfOpinion, LegalWritOfProbateWill, LegalWritOfProhibition, LegalWritOfRight, LegalWritOfScireFacias, LegalWritOfSequestration, LegalWritOfSubpoena, LegalWritOfSummons, LegalWritOfVenireFacias, LegalWritOfWarrant

10. The *GenericInstrument*[19] Class

The *GenericInstrument* class (Table XXI) is another important subclass of *GenericLegalDocument.* This class features both its own subclasses plus a union of instrument types categorized by their subject matter. Its super-classes show that instruments may be amended, attested, delivered, executed, notarized, ratified, subrogated, subscripted, and transferred. Instruments are also temporal entities, meaning they have a beginning 'effective' date time and an ending 'expiration' date time.

The superclasses for the *GenericContract* class (Table XXII) highlight the possible states for a contract, e.g., proposed, offered, accepted, declined, countered, reneged, and defaulted. The subclasses shown decompose in the ontology into contracts specific for industries, types of good, and so forth.

[18] Excluded are GenericBillOfLading, GenericBrief, GenericOrder, GenericPetition, GenericRelease, and GenericTreaty. For GenericInstrument, see Table XXI.

[19] http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF:GenericInstrument

Table XXI. *GenericInstrument* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Code	LegalCode	Format	GenericLegalDoc't	NegotiableInstr't
GoverningInstr't	GenericInstr't	Relation	TemporalThing	TestamentaryInstr
Jurisdiction	LegalJurisdiction	Format	AmendableThing	
InterestedParty	GenericActor	Contributor	AttestableThing	**Unions**
Party	GenericActor	Creator	DeliverableThing	GenericBond
SubordinateInstr't	LegalInstrument	Relation	ExecutableThing	GenericContract
			NotarizableThing	GenericDeed
			RatifiableThing	GenericLease
			SubrogatableThing	GenericWill
			SubscriptableThing	
			TransferableThing	

Table XXII. *GenericContract* Class Relations

Properties	**Range**	**Category**	**Superclasses**	**Subclasses**
Consideration	FactualStatement	Description	GenericInstr't	AmendmentAgree'tInst't
Obligation	GenericEvent	Description	AcceptableThing	BuybackAgree'tInst't
			CounterableThing	BuysellAgree'tInstr't
			DeclinableThing	ExtensionAgree'tInstr't
			DefaultableThing	**Unions**
			OfferableThing	AgencyAgree'tInst't
			ProposableThing	CivilAgree'tInst't
			RenegableThing	CommercialAgree'tInst't
				FinancialAgree'tInst't
				OwnershipAgree'tInst't
				PublicWorksAgree'tInst't
				PurchaseAgree'tInstr't
				ServiceAgree'tInst't
				UseAgree'tInst't

11. Concluding Remarks

Legal-RDF is creating an ontology useful during semantic annotation of the content of XHTML documents; during exchange of RDF documents; and during execution of ECMA software. The strengths of its ontology are found in its commitment to

(a) integration of basic markup standards (RDF, XML Schema, and XHTML) and international standards (ISO, SI, and NAICS);

(b) segregation of predicate verbs from predicate nouns;

(c) organization of all defined attributes into Dublin Core categories;

(d) adoption of a pronounced perspective for defined concrete classes;

(e) establishment of quality-laden class hierarchies; and

(f) adherence to a statement-based model for legal (document) content.

Support for these requirements are lacking or incomplete in the candidate ontologies reviewed during the initial design of the Legal-RDF ontology. For instance, DOLCE lacks (a), (b), (c), and (f); (d) is implicit in its use of the controversial perdurant/endurant model, and (e) occurs non-rigorously in its set of base classes. The SUMO and LKIF ontologies have similar profiles which also prevented their use towards the goal of the Legal-RDF ontology: to represent the *entirety* of a legal document in a manner practical to both government and industry.

Since this ontology seeks to meet the needs of legal professionals, then its scope needs to encompass all types of documents they encounter. As the Legal-RDF ontology evolves from Version 1 (which has 15,000+ terms) to the expressive models of Version 2, and as it leverages the economies of a public Wiki, this goal is both realistic and attainable.

12. Readings

DOLCE Ontologies: `http://www.loa-cnr.it/DOLCE.html`

Legal-RDF Ontologies:
Ver 1: `http://www.legal-rdf.org/` and `http://www.legal-xhtml.org/`
Ver 2: `http://aufderheide.info/lexmlwiki/index.php?title=Legal-RDF_Ontologies`

LKIF Ontology: `http://www.estrellaproject.org/lkif-core/-documentation`

McClure, J., Annotation of Legal Documents in XHTML, V Legislative XML Workshop. June, 2006. `http://www.ittig.cnr.it/legws/program.html`

Niles, I., and Pease, A. 2001. `http://projects.teknowledge.com/HPKB/Publications/FOIS.pdf` Towards a Standard Upper Ontology(`http://projects.teknowledge.com/HPKB/Publications/FOIS.pdf`). In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds, Ogunquit, Maine, October 17-19, 2001.

Xerox PARC, `http://www.parc.xerox.com/research/projects/aspectj/default.html`