

# Qualitative Geospatial Processing, Ontology and Spatial Metaphor

David Dodds  
Open-Meta Computing Inc  
Coquitlam, B.C. Canada

There are many useful circumstances where a computer is able to wield competence about qualitative geospatial concepts and data. This is accomplished through the use of ontologies whose integrated usage coupled with qualitative computer processing (“non-crisp”) provides program capabilities reminiscent of some of the conceptual metaphorical usages of natural language.

## Computer is able to Wield Competence about Qualitative Geospatial Concepts and Data

Rene Descartes' representation system of orthogonal dimensions of space, Cartesian co-ordinates, are a conceptual application of the real-number-line, yet another spatial metaphor. These and other common spatial concepts are projections of concepts into a spatial realm. There has been extensive research into use of analogy and metaphor in cognition and representation, as well as semiotics. Among these are Lakoff, Bateson, and Barwise.

Among many conceptual metaphor examples, Lakoff explains that the metaphor “MORE IS UP”, “LESS IS DOWN” is based on one's experience of UP-DOWN spatial orientation, and if one “adds more of a substance or of physical objects [pebbles, tokens, chips] to a container or pile, the level goes up”. Hence we have expressions like “My income rose last year.”, “The stock market crashed in 1929.”

Use of physical objects as counters (of goats, bales of grain, etc) long ago evolved into pebbles (calculi) in grooved boards and then sliding on wires (soroban, abacus). Physical counters were aligned, moved and otherwise operated on according to a two-dimensional spatial regime (defined by those grooves or abacus-wires). Counting became, in all cultures which used physical counters, a matter of manipulating physical objects (such as pebbles) in very particular way vis-a-vis physical space (grooves, wires, etc). Thus “concrete” space and perceptual space were used in projecting or externalizing mental functions. For millennia various cultures have used a folk or subjective spatial reference system centered on the human body. (At one point the definition of a yard was the length of the King's arm.) And as a result, we have expressions such as “just out of reach”, “within my grasp”, “a day's ride”, and comparatives based on relative height, where “daddy” is a towering giant to a baby, and maybe a bit shorter than “me” when that baby is a teenager. In these latter cases (of folk or subjective measure) we never know any exact length, distance, speed, weight and so on, they are all relative to “my subjective bodily experience”. (And none of us know the exact length of our arm, or the exact motion vector of all those people on a crowded sidewalk that we don't crash into while walking among them.)

Replacing those physical items (pebbles etc) with written symbols and a spatial operating surface, such as clay tablet, papyrus, chalk slate, or paper, cultures then invented machines which could perform (analogs of) those previous actions (operating the pebbles on the abacus-wire). We have replaced motorized gears (adding machines) with integrated circuits, bit-patterns replacing written symbols and frozen some of those physical operations on objects into a small set of machine “logical operations”. (but there are precious few of them and they are really primitive. It is as though, logically, we have to use (logical) erector sets to build the (complex knowledge) Golden Gate bridge.) Also since pebbles represented goats, bags of grain and so on there was no “zero” or “no thing” pebble, where is the zero-value bead on an abacus, what is the symbol for zero in Roman numerals? Black and shadows is the absence of light, vacuum the absence of molecules, yet linguistics (natural language) allows for explicit

representation of non-existent things in the same manner as things which do exist.

Geo-spatial concepts are those which are common to all of our thinking such as up-down, left-right, inside-outside, and so on. We can use SVG and CAD-CAM systems to represent these concepts. (Later on in this paper there is SVG code and an SVG picture. Notice that the text “UP” is placed at x="230" y="20" in that picture. Other programs can “look at” this SVG picture (examine its code actually) and “discover” that the place “UP” is associated with low values of Y-axis. By simple extension “up-ward” is the “direction toward” low-valued Y-axis values. The location for “UP” was defined by humans, the metaphorical or semiotic association of the physical label “UP” with low-valued Y-axis values can be discovered by standard data-mining practices, including those represented in PMML.)

We can use fuzzy sets applied to these systems to functionally deal with those subjective body-centered spatial notions in a computer. We can define ontologies which represent semantic knowledge about requisite domains such as space and time. Procedural attachment and other “executable logic” (Prolog, OPRS, etc) can be associated with those ontologies.

The design of OntoSensor [16], for example, “includes definitions of concepts and properties adopted in part from SensorML, extensions to IEEE SUMO and references to ISO19115. SensorML, an initiative of OGC, uses XML representation. It is not an ontology because it provides no logical or axiomatic-grounded theory to account for its conceptualization. An OWL ontology with formal definitions of GML concepts is being created by Defne et al using Protege-2000. Referencing this ontology into the OntoSensor namespace is the preferred approach in the interest of knowledge reuse and modularity.” OntoSensor uses GML: Time Primitive, GML:Time Instance, GML:Time Range and GML:Engineering CRS comprised of GML:Coordinate System, and GML:Datum.

Here are the main ontologies used by the system in this paper:

```
xmlns:FuzzyOnt="http://site.uottawa.ca/~mkhedr/FuzzyOnt#" [NewFuzzy.owl]
xmlns:time-entry = "http://www.isi.edu/~pan/damlttime/time-entry.owl#" [DAMLtime.owl]
xmlns="http://opencyc.sourceforge.net/daml/cyc#" [Opencyc.daml] (lake example follows)
```

```
<owl:Class rdf:ID="Lake">
  <rdfs:label xml:lang="en">lakes</rdfs:label>
  <rdfs:comment>A specialization of #BodyOfWater. Each instance
    of #Lake is a land-locked body of water, typically but not
    necessarily of freshwater. Two important specializations
    are #FreshWaterLake (instances of which are fresh-water
    lakes) and #InlandSea (instances of which are salt-water
    lakes). Examples include #LakeMaracaibo, #LakeWinnipeg,
    #LakeTanganyika, #LochNess-Lake, #LakeErie, #LakeTahoe,
    #CaspianSea, and #DeadSea.</rdfs:comment>
  <guid>bd58b822-9c29-11b1-9dad-c379636f7270</guid>
  <rdf:type rdf:resource="#PublicConstant"/>
  <rdf:type rdf:resource="#ExistingObjectType"/>
  <rdfs:subClassOf rdf:resource="#BodyOfWater"/>
  <rdfs:subClassOf rdf:resource="#Individual"/>
  <owl:disjointWith rdf:resource="#Stream"/>
```

</owl:Class>

Mountain and mountain range example CYC predicates follow:

<owl:Class rdf:ID="Mountain">

<rdfs:label xml:lang="en">mountains</rdfs:label>

<rdfs:comment>A specialization of #LandTopographicalFeature.

Each instance of #Mountain is a topographical region on the planet Earth of significantly higher elevation than its surrounding area. Mountains may occur individually or as

part of a chain (see #MountainRange). Instances of #Mountain include #MountWhitney, #DiamondHead-Mountain, #MountKosciusko, #AyersRock, and #MountOlympus.</rdfs:comment>

<guid>bd58ce0d-9c29-11b1-9dad-c379636f7270</guid>

<rdf:type rdf:resource="#PublicConstant"/>

<rdf:type rdf:resource="#ExistingObjectType"/>

<rdfs:subClassOf rdf:resource="#PartiallyTangible"/>

<rdfs:subClassOf rdf:resource="#LandStuff"/>

<rdfs:subClassOf rdf:resource="#Individual"/>

<rdfs:subClassOf rdf:resource="#LandTopographicalFeature"/>

<rdfs:subClassOf rdf:resource="#TopAndBottomSidedObject"/>

<rdfs:subClassOf rdf:resource="#Protrusion"/>

<rdfs:subClassOf rdf:resource="#Individual"/>

</owl:Class>

<owl:Class rdf:ID="MountainRange">

<rdfs:label xml:lang="en">mountain ranges</rdfs:label>

<rdfs:comment>A specialization of #LandTopographicalFeature.

Each instance of #MountainRange is a natural group of mountains. Instances of #MountainRange include the

#RockyMountains, #Andes-Mountains, and #Alps-Mountains.</rdfs:comment>

<guid>bd58e52f-9c29-11b1-9dad-c379636f7270</guid>

<rdf:type rdf:resource="#PublicConstant"/>

<rdf:type rdf:resource="#ExistingObjectType"/>

<rdfs:subClassOf rdf:resource="#GeographicalRegion"/>

<rdfs:subClassOf rdf:resource="#Mob"/>

<rdfs:subClassOf rdf:resource="#Individual"/>

<rdfs:subClassOf rdf:resource="#LandTopographicalFeature"/>

<rdfs:subClassOf rdf:resource="#Individual"/>

</owl:Class>

<owl:Class rdf:ID="Grassland">

<rdfs:label xml:lang="en">grassland</rdfs:label>

<rdfs:comment>Grassy land with deep, rich soil and few trees or shrubs.</rdfs:comment>

<guid>be0100a3-9c29-11b1-9dad-c379636f7270</guid>

<rdf:type rdf:resource="#ClimaticTerrainType"/>

<rdf:type rdf:resource="#ProposedPublicConstant"/>

<rdf:type rdf:resource="#PublicConstant"/>

<rdf:type rdf:resource="#ExistingStuffType"/>

<rdfs:subClassOf rdf:resource="#GeographicalRegion"/>

```

<rdfs:subClassOf rdf:resource="#Plain-Topographical"/>
<rdfs:subClassOf rdf:resource="#Individual"/>
</owl:Class>

```

```

<owl:Class rdf:ID="Tree-ThePlant">
  <rdfs:label xml:lang="en">trees</rdfs:label>
  <rdfs:comment>A specialization of #Plant-Woody. Each instance
    of #Tree-ThePlant is a tall woody plant (typical mature
    specimens are usually taller than a person or a bush),
    generally having a branching form overall, and with roots in
    the ground, a trunk, and the branches having numerous leaves
    exposed to the sky. Notable specializations of
    #Tree-ThePlant include #OakTree, #MapleTree,
    #GiantSequoia, and #BananaTree.</rdfs:comment>
  <guid>bd58cf75-9c29-11b1-9dad-c379636f7270</guid>
  <rdf:type rdf:resource="#OrganismClassificationType"/>
  <rdf:type rdf:resource="#PublicConstant-DefinitionalGAFsOK"/>
  <rdf:type rdf:resource="#PublicConstant-CommentOK"/>
  <rdf:type rdf:resource="#PublicConstant"/>
  <rdf:type rdf:resource="#ExistingObjectType"/>
  <rdfs:subClassOf rdf:resource="#Plant-Woody"/>
  <rdfs:subClassOf rdf:resource="#Individual"/>
</owl:Class>

```

```

xmlns:Context = "http://site.uottawa.ca/Context#" [Context.daml_oil]
rdfs:range rdf:resource="#open-meta.com/daxsvg" [DAXSVG.rdfs]
xmlns="http://reliant.tekknowledge.com/DAML/SUMO.owl#" [IEEE SUMO.123]
xmlns="http://loki.cae.drexel.edu/~wbs/ontology/2004/09/
http://ontology.buffalo.edu/bfo/Geo.pdf [Geo geographical ontology, SNAP, SPAN,
Field]
xmlns="http://loki.cae.drexel.edu/~ontology/2004/09/ogc-gml#" [OGC-GML.OWL]

```

A simplified schematic of ontology which is referenced for basic knowledge about all physical things.

SVG TREE

```

  THING
  /   \
VIRTOBJ  PHYSOBJ
          |
        PHYSPROPERTIES:
          LENGTH  L
          WIDTH   W
          VOLUME  V
          SIZE    S
          COLOUR  C
          MASS    M
          WEIGHT  W
          SMELL   O

```

("SMELL" BY GAS SPECTROMETER, OR BY JUST COMPARING WITH A NAMED OBJECT,

(SMELLS) LIKE TAR OR PINE NEEDLES, CONCRETE.)

In this presentation we see a spatial ontology, using RDFS, with associated (Java) procedures, which implements an example of spatial tacit knowledge including how to meta-program usage of fuzzy sets to represent qualitative geo-spatial interests.

What sorts of things does that knowledge provide, well for example, when an object is recognized as two-dimensional, by means of “inheritance” etc the system is able to “realize” tacit associations, such as the object is likely bounded, it likely has an inside and an outside, an area, if it's a “real” object it likely has a colour and perhaps a texture (both visually and physically). Since part of the basic spatial (tacit) knowledge of the system is a set of naïve-physics items (like gravity, such as might exist in a computer game) it is possible for the system to derive some (admittedly somewhat simple) expectancies, such as water droplets falling out of the sky, and then forming various water conglomerations of various magnitudes and characteristics (such as “flowing” “downhill”).

### **What goes up must come down, but why?**

The gravity naïve-physics might be informed by code like:  
<http://www.jmckell.com/generalgravity.html>

The equation for the force of gravity between two objects:

$$\text{gravity} = \frac{\text{mass1} * \text{mass2}}{\text{distance}^2} * k$$

Here is some general code which calculates gravitational attraction based motion of two bodies.

```
--distance components
```

```
distX = (moh) - x
```

```
distY = (mov) - y
```

```
--pythagorean theorem to get distance
```

```
dist = sqrt(power(distX,2) + power(distY,2))
```

```
--put a lower limit on dist
```

```
if dist < sp.width then dist = sp.width
```

```
--calculate gravity, x & y components
```

```
grav = mass * moMass / power(dist,2) * 5
```

```
xGrav = grav * (distX/dist)
```

```
yGrav = grav * (distY/dist)
```

```
--accel (with mass), velocity, position
```

```
xAccel = xGrav / mass
```

```
yAccel = yGrav / mass
```

```
xVelo = xVelo + xAccel
```

```
yVelo = yVelo + yAccel
```

```
x = x + xVelo
```

```
y = y + yVelo
```

These calculations might constitute the system's knowledge of gravity, providing it with the ability to compute "expectations" of the behaviour of objects affected by gravity. An example would be the expectation that an unsupported object would fall "to the ground". And in a certain manner. Other than support by wind and aero-dynamic lift unsupported objects in the air would be expected to fall ("to the ground"), even drops of water, or a thrown rock.

An advanced system would be able to intake measurements from instruments, build a model which explains the data and use those models, such as vertical plant growth, to develop expectations which are incorporated into the general intelligence of the overall system.

## **Water and Heat and Steamy Jungles, Oh My**

Natural language-like statements assist XML-based data structures in the description of and utilization of geo-spatial information, especially for those who have substantially greater facility with English statements and sentences than with XML (GML) and its editor and parsing programs. Natural language can assist this former population to create and interact with ("consume") geo-spatial data. One of the ways this can be done is by using computer programs which work with data-bases and with representations such as GML. The data and a representation of the intention of the user re the data are used to create English-like statements. In the opposite direction English statements produced by human users are parsed by computer and translated into XML GML (and perhaps meta-data). The level of English sentence and statement would have to be relatively unsophisticated but would provide a broader range of geo-spatial data users than those restricted to having knowledge of XML or at least GML savvy text editing tools. However a tool which is capable of parsing GML data structures does not in itself have capabilities built-in which can take the products of that parsing and expressing it as (decent) English.

Suereth has published a book on natural language processing (which provides C language source code) which implements the processing (of NL) which he discusses in his book. A very brief snip of his code follows:

```
/*
*****
*/
/* CONVERSE.C
*/
/*
*/
/* Text Processor Copyright (c) 1996 Russell Suereth */
*****

/*
*****
*/
/* This routine processes pragmatic knowledge.
*/
/* Display a paragraph analysis line with the subject, action and
*/
/* place of the current input sentence.
*/
*****
void pragmatic_display_detail()
{
/* If the current input sentence has a subject, then display
*/
/* the subject.
*/
if (strlen(semantic[sentence].sem_subject) > 0) {
if (semantic[sentence].subject_type != SUBJECT_NAME)
semantic[sentence].sem_subject[0] =
(char) tolower(semantic[sentence].sem_subject[0]);
}
```

```

    printf("%-13.13s", semantic[sentence].sem_subject);
} else
    printf("%-13.13s", "-");

/* If the current input sentence has an action, then display */
/* the action. */
if (strlen(semantic[sentence].sem_action) > 0) {
    printf("%-13.13s", semantic[sentence].sem_action);
} else
    printf("%-13.13s", "-");

/* If the current input sentence has a place, then display */
/* the place. */
if (strlen(semantic[sentence].sem_place) > 0) {
    printf("%-13.13s", semantic[sentence].sem_place);
} else
    printf("%-13.13s", "-");

sentence++;
return;
}

```

One can see that a certain range of English sentences can be parsed by a system like his and translated into corresponding XML (GML). His book does not specifically focus on geo-spatial information, nor statements about it. He does explain quite clearly how to parse syntax of English and how to extract semantic, that is, meaning information. (A parser alone does not provide the meaning of a sentence.)

The information recognition and extraction capable by a system such as Suereth's allows a computer to functionally process a statement such as "A huge tree is smaller than a small mountain." the information found is "noun phrase (subject) IS something" where the something is "smaller than a small mountain". Since his system is able to recognize and handle expressions of the form "phrase comparison term phrase" it determines "(is) smaller than" to be equivalent to the mathematical operator "<" (less than). Trees are geo-spatial objects which a (geo) database would contain instances of, along with their respective dimensions such as height, size, etc. A quite simple query of such a database could glean the shortest, tallest and median height for (all) trees (in the data base). The database is the best available version of human "experience" with trees in the world. Likewise in the database there are elements of the "mountain" geo-spatial classification, with their elevation and horizontal-extents or characterization. It is a simple mathematical comparison to see that the range of tree sizes not only does not overlap that of the range of mountain sizes but also that the range of tree sizes is in all cases less than the range of all mountain sizes and hence the computer could determine that our English statement about trees and mountains is true or "correct". "Huge" is an adjective and can be treated as a hedge computation as can "small" (huge tree, small mountain). Lotfi Zadeh describes how to compute hedges. Those interested in the details are encouraged to look at material on the open-meta.com website. [huge tree is the upper or larger values in the range of tree values in the database and small mountain is the lower or smaller values in the range of mountain's heights or sizes.]

Key among the many things that need to be in place for such a system to work is that of context. The terms used in general natural language are embedded in a culture (the speaker's "experience reference") and in a localized context which defines a situation. In this way the terms are given (tacit) nuance by

virtue of being transmitted in such a context.

In the case of the statement, “A huge tree is smaller than a small mountain.”, parsing provides syntactical information including phrasal recognition of “is smaller than” which with modest semantic processing can be turned into the equivalent of “ $A < B$ ” and a context of numeric arguments. (A,B are expected to be quantities.) Also in the context is the notice that “smaller” is an example of type “size”. Size is a dimensional aspect of spatial things. The basic spatial knowledge of the system as spoken of earlier in the presentation conveys that “size” (of thing) is nuanced; 1) magnitude of three-dimensional extents, 2) magnitude of two-dimensional extents, 3) “length”, 4) “height”, 5) “width”, 6) “mass”.

The phrase “A huge tree” is bound to A and the phrase “a small mountain” to B. The phrases themselves are strings and the system is aware that they should be quantities. Syntactic analysis of A provides that “tree” is a noun, and “huge” an adjective. The system knows that a noun is the name of a person, place or thing. Since it might be a known-thing by virtue of being in a geo-data-base or an ontology an attempt is made to look it up. In the case where “tree” is in the db then an organized search of data about the object can be made. Because of the contextual knowledge that a “size” feature is sought after any db entries in “tree” for “magnitude of three-dimensional extents, or magnitude of two-dimensional extents, or height, or mass” are used as potential numeric arguments. (ex. height=50) Syntactic analysis of B provides that “mountain” is a noun, and “small” an adjective. A db search for “mountain” objects and any relevant (nuanced) data associated with them locates the same kind of “magnitude of three-dimensional extents, or magnitude of two-dimensional extents, or height, or mass” information regarding “tree”. By lining up the same (attribute) type for each object (“tree” and “mountain”), say “magnitude of three-dimensional extents” the computer is then in a position to implement  $A < B$ . (Specifically this comparison is

IF magnitude of three-dimensional extents of trees  $<$  magnitude of three-dimensional extents of mountains THEN statement is “accurate” (and “true”).

The comparison could also be of the form:

IF height of [trees]  $<$  height of [mountains] THEN statement is “accurate” (and “true”).

IF mass of [trees]  $<$  mass of [mountains] THEN statement is “accurate” (and “true”).

In the case of the statement, “A huge tree is bigger than a small mountain.”, the same computer based reasoning would find the statement “not accurate” (and by implication “false”).

The case of the statement, “A big dog is smaller than a small house.”, the same computer based reasoning would find the statement “accurate” (and by implication “true”). The database would need contain information about “dog” and “house” objects (nouns), in the same way it would about “tree” and “mountain” objects.

The “computer code knowledge” to perform the activity above may be provided in the form of PMML, which is an XML name space. Also procedural knowledge could be provided via PMML to perform XBL operations on SVG representations of objects (like the trees and mountains) so as to show visual illustrations of things being processed. These visualizations could well have SVG meta-data embedded within.

Geo-spatial data represented in GML may be parsed and inserted into English language templates which state the same information but embedded in an English-like presentation or form. This English provides for a broader spectrum of user.

### Oceans Fall from the Sky

To provide a measure of inference capabilities using a GML representation of the real world it is possible to include naïve physics like in computer games to the system. A computer usable notion like gravity is an example. There are a number of facilities used in computer games, like gravity (gravitational effects on objects), that provide useful tools of inference in the geo-spatial milieu. Gravity and mass are computer representable concepts that allow a computer to predict (and, thereby, expect) that rain falls down(ward), that rivers and streams flow downward and not upward (without “help”). It allows the computer to predict, and therefore expect, that rainfall causes streams, rivers, lakes, oceans. In a way these things (streams etc) can be seen to be (ontologically) “inherited” from rainfall. Perhaps it would not be entirely surprising that steamy jungle or forest would be predicted to occur in hot equatorial regions where there is water (and trees), given such inheritance and inference computations.

(Take a look at the explanation further ahead for how Near() works. Far, AtRight, AtLeft, Front, Beside, Behind, Above, Below, Big, etc all use the same mechanism, with some different parametric values. Crisp axes data is converted into a grade of membership value (and then possibly into linguistic form like “somewhat near”) by means of continuous functions, like:

$$g(x) = \exp - \left( \left( \frac{x - k(1)}{k(2)} \right)^2 \right).$$

and graphs of very useful functions:

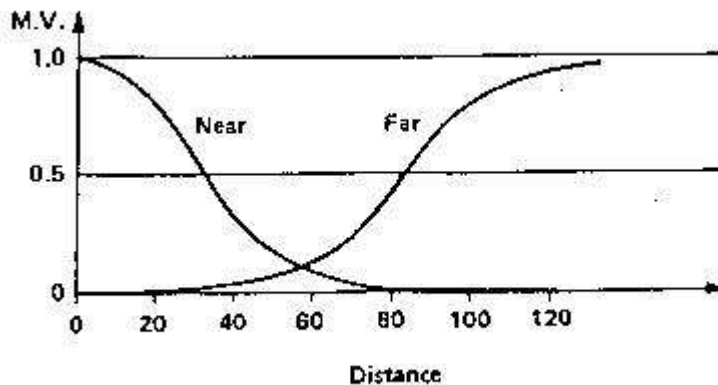


Fig. 1.

$$f(x) = \frac{1}{2} + \frac{1}{\pi} \tan^{-1} \left( \frac{x - k(1)}{k(2)} \right).$$

## The sciences of astronomy and physics have long used conceptual metaphor

The sciences of astronomy and physics have long used conceptual metaphor productively, in such terms as “*black hole*”, “*event horizon*”, “*gravity well*”, “*curved space*”, “*time dilation*”, “*light year*”, “*dark matter*”, “*red giant*”, “*white dwarf*”. Computing Science has spatial terms such as (aspect oriented programming's) “*cross-cutting concerns*”, “*pointcuts*”, “*join points*”, “*aspects*”.

The same technologies which provides the mechanisms for such term usage is available to geo-spatial processing. Productive conceptual metaphors can be used functionally on computers. This may be done by employing ontologies which provide requisite background knowledge to the system and qualitative logic processing. For example, a metaphorical comparison between two geo-spatial entities is qualitative and able to arrive at (useful) data when often strict-equality and simple scalar comparison either outright fails or returns a possibly nonsensical answer. An everyday example of a metaphorical comparison between two spatial entities is “The fallen apple lay at the foot of the tree.”, “The river ran past the foot of the mountain.” The metaphorical usage / comparison is human foot (being located at the lower location of the human) and foot of tree and mountain (being at a (corresponding or analogous) lower location of these), neither trees nor mountains have anything like a human foot, it is the relative location on the body of the foot that makes the metaphor / analogy, not its presence upon trees and mountains.

## XML RDFS spatial ontology

Code based examples follow. First we see an XML RDFS spatial ontology developed by David Dodds. (It also has a brief, very simple, time ontology rolled into it.) A Context mechanism is associated with the ontology, a crucial aspect.

```
<!-- Copyright 2001 - 2005 David Dodds
```

```
Context {  
change of state,  
occurrence of event,  
occurrence of particular pattern of states and or events,  
occurrence of a sequence (detected via HMM Hidden Markov Model),  
or graph (such as XGMML)  
}  
-->
```

```
<rdf:RDF xml:lang="en"  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">  
<rdfs:Class rdf:ID="SvgEntity">  
<rdfs:comment>The class of SVG entities,  
referenced by their id in the SVG code.  
THIS (DAX) schema is modeled on the schema at  
http://www.w3.org/2000/01/rdf-schema#Resource.  
THIS schema is called daxsvg-schema-rdf.xml.
```

It is more extensive than the w3 model, also this one has procedural attachment.

```
</rdfs:comment>
<rdfs:subClassOf rdf:resource="http://www.open-meta.com/2000/01/
rdf-schema#Resource"/>
</rdfs:Class>
<rdf:Property ID="Near">
<rdfs:comment>has a degree of nearness (by value).  $g_1(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Far">
<rdfs:comment>has a degree of farness (by value). complement of near,
 $1 - g_1(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Large">
<rdfs:comment>has a degree of largeness (by value).  $g_2(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Small">
<rdfs:comment>has a degree of smallness (by value). complement of large,
 $1 - g_2(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Big">
<rdfs:comment>has a degree of largeness (by value).  $g_2(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Little">
<rdfs:comment>has a degree of smallness (by value). complement of large,
 $1 - g_2(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Fast">
<rdfs:comment>has a degree of quickness (by value).  $g_3(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Slow">
<rdfs:comment>has a degree of slowness (by value). complement of fast,
 $1 - g_3(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
```

```

<rdf:Property ID="Before">
<rdfs:comment>has a degree of occurance (by value) at time
values monotonically decreasing from a value t1, on the T(x)
timeline. g31(x). a named ordered collection may be used instead
of the timeline.</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="During">
<rdfs:comment>has a degree of occurance (by value) at time
values between t1 and t2, on the T(x) timeline. g32(x). a named
ordered collection may be used instead of the timeline.</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="After">
<rdfs:comment>has a degree of occurance (by value) at time
values monotonically increasing from a value t2, on the T(x)
timeline. g33(x). a named ordered collection may be used instead
of the timeline.</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Often">
<rdfs:comment>has a degree of quickness (by value). g4(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Seldom">
<rdfs:comment>has a degree of infrequency (by value). complement of often,
1 - g4(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Simultaneously">
<rdfs:comment>has a degree of multiplicity (by value). g5(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Egressive">
<rdfs:comment>has a degree of departureeness (by value). g6(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Ingressive">
<rdfs:comment>has a degree of arrivedness (by value). complement of
egressive, 1 - g6(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />

```

```

</rdf:Property>
<rdf:Property ID="Changing">
<rdfs:comment>has a degree of changingness or quickness (by value).
g7(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Static">
<rdfs:comment>has a degree of unchangingness (by value). complement of
changing, 1 - g7(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Touches">
<rdfs:comment>has a degree of same location (by value). same location as
object's x,y g14(z)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="AtRight">
<rdfs:comment>has a degree of to the right (by value). g15(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="AtLeft">
<rdfs:comment>has a degree of to the left (by value). g16(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Center">
<rdfs:comment>has a degree of [at the] centerness (by value). g8(x).
SVG maxx - minx / 2, maxy - miny / 2</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Periphery">
<rdfs:comment>has a degree of outerness (by value). complement of center,
1 - g8(x)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Containz">
<rdfs:comment>has a degree of containingness (by value). uses center g8(x).
</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Lacunarity">
<rdfs:comment>presents a curve (by value). Its fractal lacunarity L. g9()

```

```

</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="FractalDimension">
<rdfs:comment>presents a curve, surface, or volume (by value).Its fractal
dimension D. g10()</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Significance">
<rdfs:comment>has a degree of importance (by value). Product of relevance
g11() and novelty g12()</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Top">
<rdfs:comment>absolute marker, maximum y value in 2D reference system. (SVG
miny, origin upper left)</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Bottom">
<rdfs:comment>absolute marker, minimum y value in 2D reference system. SVG
maxy</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Left">
<rdfs:comment>absolute marker, minimum x value in reference system.
SVG minx</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Right">
<rdfs:comment>absolute marker, maximum x value in reference system.
SVG maxx</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Front">
<rdfs:comment>absolute marker, 0 degree angle in reference system
x axis</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Back">
<rdfs:comment>absolute marker, 180 degree angle in reference system
x axis</rdfs:comment>

```

```
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Above">
<rdfs:comment>absolute marker, 90 degree angle in reference system
z axis</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Below">
<rdfs:comment>absolute marker, -90 degree angle in reference system
z axis</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Inside">
<rdfs:comment>has a degree of insideness (by value). g27(x). inside is the
containee perspective of containz</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="Self">
<rdfs:comment>reference to outer extents of (containz) subjective reference
system</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="colour">
<rdfs:comment>numerical representation of colours using rgb model
</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="morphology">
<rdfs:comment>description vector defining shape of something. includes
fractal D and L.</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="size">
<rdfs:comment>a second order descriptor which is a relative measure of
object area or volume. The SVG canvas information contained in the SVG
element of a picture is usually the basis of the relative comparison.
This is used as the reference frame. The box method of fractal dimension
may also be used to obtain a measure of size.</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="location">
```

```

<rdfs:comment>a second order descriptor which uses the Near, Far, AtRight,
AtLeft, InFrontof, Above, Below, Behind, Inside, Containz, Beside,
Periphery, Center, Touches predicates as necessary and is represented
either by a property list representation of the resulting findings or by
an XGMML context graph structure as needed.</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="mobility">
<rdfs:comment>description vector defining position change of something.
includes fractal D and L.</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
<rdf:Property ID="time">
<rdfs:comment>description vector defining time count of something relative
to a beat source; or sequence. The default source of time is the computer
system clock (i.e. timestamp).</rdfs:comment>
<rdfs:range rdf:resource="#SvgEntity" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
</rdf:RDF>

```

Some items from the (UnivOtt) Context ontology, which is used in modified form:

```

<daml_oil:ObjectProperty rdf:ID="actionExpectedDuration">
<daml_oil:domain rdf:resource="http://site.uottawa.ca/Context#ActionProfile"/>
<daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#duration"/>
</daml_oil:ObjectProperty>
<daml_oil:ObjectProperty rdf:ID="precondition">
<daml_oil:domain rdf:resource="http://site.uottawa.ca/Context#Application"/>
<daml_oil:domain rdf:resource="http://site.uottawa.ca/Context#Resource"/>
<daml_oil:range rdf:resource="http://www.daml.org/2004/03/daml+oil#Thing"/>
<daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#anyType"/>
</daml_oil:ObjectProperty>

```

David's comment: One of the components of my notion of "situatedness"; the location of any actors involved in the current context. Note that the ontology defines actorLocation but is not able to "get" a (location) value for it. That is up to the (executable) logic which exists and is used outside the ontology. Later in this paper we see code that does this. The uottawa.ca/Context ontology does not provide such code.

```

<daml_oil:ObjectProperty rdf:ID="actorLocation">
<daml_oil:domain rdf:resource="http://site.uottawa.ca/Context#Actors"/>
<daml_oil:range rdf:resource="http://site.uottawa.ca/Context#Location"/>
</daml_oil:ObjectProperty>

```

David's comment: One of the components of my notion of "situatedness"; the identification of exactly what are the constituents of the context. Note that the ontology defines Physical but is not able to "get" a value for it. That

is up to the (executable) logic which exists and is used outside the ontology. Program logic, in a Java program called Physical, provides the information for the ontology slots like Physical (the ontology item) such that all of the ContextFeatures in the Context are determined. (i.e. an OWL version Context ontology would have a slot for Context.ContextFeature. Physical and the slot would be populated with a value by the Java code).

```
<daml_oil:Class rdf:ID="Physical">  
<rdfs:subClassOf rdf:resource="#ContextFeature"/>  
</daml_oil:Class>
```

David's comment: Here we see how the ontology conveys to the computer the distinction between an actor who is a person (a human), and an actor which is not a human (software or a machine). The distinction is made via the disjointWith part of the statement below. There are scenarios where the expected sophistication of an action can be estimated by knowing what kind of actor is performing the action. Through inferencing then the system would be able to determine that evidence of stealth be sought when the actor is not an agent. Such inferencing on the Context provides the means to add new otherwise tacit items to the Context, thereby making it more explicit or concrete in its coverage of the situation.

```
<daml_oil:Class rdf:ID="Agent">  
<rdfs:subClassOf rdf:resource="#Actors"/>  
<daml_oil:disjointWith rdf:resource="#Person"/>  
</daml_oil:Class>
```

David's comment: Notice with these next three predicates that knowledge of Actors and Roles (which is based on their appearance in the resource parts of the daml:oil expression of the knowledge-base) is used to detect and define the constituency of the context (under construction via use of these predicates). A context of the SAW Situation Awareness type is populated / constructed incrementally / dynamically by utilizing (the (usually Java-based or XSLT) action code associated by name with) such predicates, and the further predicates whose relevancy is based on discovery (of, in this case, other Actor and or Role related ontological knowledge association.) Simple-relevancy is discovered by searching the predicate base for (rdf) resources which are Actors or Roles, in this case. One might here liken Actors and Roles to context focus (POV), which may be used to inform the discovery process. (Which might be performed by an XSLT search of the predicate base looking (in a predicate's rdf:resource) for "Actor" and "Role".)

```
<daml_oil:ObjectProperty rdf:ID="actorRole">  
<daml_oil:domain rdf:resource="http://site.uottawa.ca/Context#Actors"/>  
<daml_oil:range rdf:resource="http://site.uottawa.ca/Context#Role"/>  
</daml_oil:ObjectProperty>  
<daml_oil:ObjectProperty rdf:ID="thingName">  
<daml_oil:domain rdf:resource="http://site.uottawa.ca/Context#ThingRole"/>  
<daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#anyType"/>  
</daml_oil:ObjectProperty>  
<daml_oil:ObjectProperty rdf:ID="hasCurrentAction">  
<daml_oil:range rdf:resource="http://site.uottawa.ca/Context#Action"/>  
<daml_oil:domain rdf:resource="http://site.uottawa.ca/Context#Actors"/>
```

</daml\_oil:ObjectProperty>

The appearance (presence) of the predicate "hasCurrentAction" in the current context is used by my system to activate the action-code of the same name (i.e. hasCurrentAction()). A simple looping Java-based or XSLT-based scan of the current context ontology (its data structure) can be used to detect any new entry in the context such as the initial or new appearance of the predicate "hasCurrentAction" and then run the program of the same name, hasCurrentAction() function which examines (other) data structures for information relevant to detecting - determining what currently any identified actors are doing. (In the example used in this paper, a diagram animation, anim01.svg, there are three actors and they are identified as such because they show change in location ("movement") in the animation. The function hasCurrentAction() need only (in this case) find the "animate" SVG term in the SVG diagram picture (i.e. in the SVG code). Later in this paper we see how the SVG program is examined for other information such as tacit spatial relationships. That is also driven by ontological and contextual knowledge.

Next we see an XML RDFS element (Near( ) ) from the author's spatial ontology. Because it is XML it is compatible for processing with XML based GML material. Following this spatial ontology excerpt are function representations which show the meaning of (the function) "g1 (x)" which occurs in the comment of the Near( ) predicate. The Near( ) predicate (shown in the RDFS) has concomitant code (Java) which implements the calculation Near (as g1(x)) which returns a scalar value, not a boolean, which represents the degree of "nearness" found. The "degree" calculation is "*situated*", that is, it is done within a context.

Spatial metaphors and analogies require availability of both "spatial knowledge" and the capabilities of manipulation and operation which characterize meta-processing of "ideas" and elements of the physical world. The "spatial k" is comprised of "facts", which amount to static data assertions, and of "changes", which amount to procedural information and "computer code".

Spatial knowledge is often based on subjective use of one's own body as a source of measurement value reference. For example, the height of a person being "my height", or "a little shorter than me", etc. Distances measured relatively as "arm's length", "within reach", "a couple of steps away". Sizes measured as "much bigger than me", "half my size" and so on. Humans have the ability to use a non-self origin as the measure but yet employ the same projective mechanism such as in "a day's ride", a league, a mile, a light-year.

Near is "analog sense" not so much a precise engineering measurement. Systems using the latter are previously copiously described elsewhere. Being able to functionally provide a computer with some of the "spatial sense" of humans allows them to be used in a more subjective less "engineering" way in that conceptual terms more akin to daily linguistic use may be employed and yet remain computationally compatible with formalisms like GML. By coupling representations of elemental spatiality ("up", "front", "container" and so on) and representations of how to perform particular actions to achieve goals ("how to" knowledge) the computer can compute approximations to the "analog sense". Part of the solution is to use so-called "fuzzy sets" and their derivations. The fuzzy functions by their nature are embedded in a context. This presentation provides some concrete examples of such context and its use with fuzzy terms.

Part of the base knowledge the system has regarding spatiality is that of constituency of objects in the world. Real objects are known (to the system) as having such attributes as: morphology (length, width, height), mass or weight, surface texture, colour / visual pattern / visual texture, temperature, general plasticity, general albedo and some others. Virtual objects, which are described “things” like (ideal) spheres, don't actually exist in the real world. They may be processed AS THOUGH (real) objects. This is what science and technology does via functional metaphors and analogies.

The example of Near( ) shown in this section is the computation of a scalar value which represents the grade of membership of a fuzzy function depicting the Near ( ) predicate. The basic version of Near ( ) takes the coordinate location of two objects, represented via a point each. Initially this point is treated as an origin and is typically the centroid of each object. We see how the calculation “adapts” when the objects are different from the default assumption. All of these calculations are done within the scope of a context which informs the calculation details.(and hence provides “adaptability”).

Part of the context is this: The two geo-spatial objects involved have origins located at  $x_1, y_1$ ; and  $x_2, y_2$ , respectively. These are formed into the centroidsdistance which informs the value of  $x$  in the function  $g_1(x)$ . Hence the computer is able to process the conceptual / visual metaphor of “nearness” (between object1 and 2) and instead of merely reporting near yes or near no, as would a strict equality / first order logic ontology system, it returns a scalar which can be processed within the context into a linguistic expression, such as “sort of near”, or “quite near”, or “not at all near”. Calculation of “hedges” such as those is shown in examples of code at [open-meta.com](http://open-meta.com).

```
<rdf:Property ID="Near">
<rdfs:comment>has a degree of nearness (by value).  $g_1(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#www.open-meta.com/2001/Near" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>
```

$g_1(\text{centroidsdistance})$

$\text{centroidsdistance} = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

$\text{Near} = g_1 = 1 - (1/2 + 1/\text{PI} * \arctan((\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}) - k(1)/k(2)))$

“(Near( $x_1, y_1, x_2, y_2$ ))”

(Near [ $g_1(\text{centroidsdistance})$ ]) )

$k_1$  and  $k_2$  are the “center” and slope of a function used to perform the fuzzy mapping (of a crisp range into grade of membership. See Zadeh)

An example of a boolean determination of Above, is shown next, in Java. It assumes the  $x=0, y=0$  origin is in the upper left corner of a 2D surface; and that object 1 location is represented via  $x_1, y_1$ ; object 2 by  $x_2, y_2$ .

```
public boolean Above( int y1, int y2 )
{
    if ( y1 < y2 )
```

```

    {
        return true; /* obj x1,y1 is above obj x2,y2 */
    }
    else
    {
        return false; /* obj x1,y1 is not above obj x2,y2 */
    }
}

```

If  $x=0$ ,  $y=0$  is at the lower left corner (as it is in some graphics systems) then the function would need to be like:

```

public boolean Above( int y1, int y2 )
{
    if ( y1 > y2 )
    {
        return true; /* obj x1,y1 is above obj x2,y2 */
    }
    else
    {
        return false; /* obj x1,y1 is not above obj x2,y2 */
    }
}

```

When the reference space is 3D then Above() is about the Z axis, which is orthogonal to the computer screen or paper output, screen level being  $z = 0$ .

```

public boolean Above( int z1, int z2 )
{
    if ( z1 > z2 )
    {
        return true; /* obj x1,y1,z1 is above obj x2,y2,z2 */
    }
    else
    {
        return false; /* obj x1,y1,z1 is not above obj x2,y2,z2 */
    }
}

```

```

public real Above( int z1, int z2 )
{
    if ( z1 > z2 )
    {
        return (near(abs(z2 - z1), contextz));
    }
    else
    {

```

```

        return 0;
    }
}

```

$g1(\text{centroidsdistance})$  // ie a 3D centroidsdistance  
 $\text{centroidsdistance} = \sqrt{(x2-x1)^2+(y2-y1)^2+(z2-z1)^2}$   
 IF  $\sim(\text{Small}(r1 + r2, \text{centroidsdistance}))$  THEN  $\text{centroidsdistance} -= r1 + r2$   
 $\text{Far} = g2 = (1/2 + 1/\text{PI} * \arctan(\text{centroidsdistance}-k(1)/k(2)))$   
 $\text{Near} = g1 = 1 - g2$   
 $(\text{Near}(x1,y1,z1,x2,y2,z2))$   
 [If the "extents" ((spherical radii)  $r1$  and  $r2$ ) of the two objects being compared is a significant proportion of the separation of the centroids then the effective centroid is moved to the outer edge (extent of closest approach) of each object, on the line joining the centroids. Two trees for example.]

$\text{Above}(\text{Near}(x1,y1,z1,x2,y2,z2))$   
 $\text{Above}(\text{Far}(x1,y1,z1,x2,y2,z2))$   
 [Near and Far are direction insensitive, Above is what gives the "direction", ie the y or z axis depending on the dimensionality of the reference space. Near and Far provide a qualitative indication of "how much" above, as opposed to FOPL's T or F.]

In this system we can think of "above" as being "more up" than that which it/they are compared with. If A is above B, A is more up than B. Up is in a spatial relationship with top, itself related to the subjective (location of one's) human head. (standing pose assumed)

When there are two objects whose centroids are separated by centroids-distance (the "Cartesian distance") they are computed to be Near( ) each other with a "grade of membership" value. (a scalar). The abscissa of the function ( $g1$ ) which does the mapping has a range equal to 0 : base abscissa. Initially, at least, the base abscissa is made equal to the x-axis length of the containing SVG field. (Calculations are grounded in SVG coordinate system, as a means of reference for definition of the spatial primitives. (axes, "up", etc))

When the two objects are not represented well via points, which is when they have a non-trivial extent from the point used as centroid, then they are approximated via a sphere centered on the centroid having a radius  $r$ . ( $r1$  for object 1 and  $r2$  for object2). If  $R1 + R2$  is NOT (small(  $R1 + R2$  )) in the context of (the value of) centroids-distance, which means that the "extents" or "size" of the objects is not a trivial fraction of the distance which separates the centroids of the two objects then the Near calculation must take into account the "extents" of the two objects in its calculation of separation (vis-a-vis Near ( )). The first approximation to this is the adjustment of the value of the centroids distance by subtracting  $R1 + R2$  from it. This calculates Near( ) based on separation of the two objects by the two points on their outer extremes which are each closest to the other object.

When a sphere is not a good approximation of the morphology of an object then the line which joins the centroids of the two objects is used to determine the closest point on the periphery of the two objects and that separation is used to calculate Near( ).

For example, Near( ) of two trees can be calculated representing the trees as vertical cylinders (with possibly a spherical top) and calculating Near( ) as a function of the separation of the two trunks. If the

arboration at the top of the trees is significant (with respect to the magnitude of the trunks) then the Near( ) calculation uses the separation of the two trees between the point on each arboration sphere which most closely approaches the other arboration sphere. These are common sense visual approximations which humans make. The context is set by using the magnitudes of each object (ie the two trees) and using either their trunks or their arboration instead if that is a significant magnitude compared with that of the trunk. If the two objects were mountains instead of trees Near( ) would use morphological knowledge relevant to mountains vs that of trees. In this way the context is informed by the morphology of the comparators (trees, mountain, stars) and hence very different distances have a similar Near( ) value [like “somewhat Near”] because Near( ) and its kin are situated! Trees that are “somewhat Near” are physically much closer together than mountains that are “somewhat Near”, and stars that are “somewhat Near” to one another are hugely farther apart than any mountains ever will be. This is reflected in our every day use of “near” in natural language because there is tacit context there and also tacit “background knowledge” too.

GML data is shown in example of qualitative geospatial processing. Computer searches of GML data such as “Find all the smallish shallow water that is somewhat near to a gully or fairly small mountain.” may be of interest.

The computer parses the statement and finds, piecewise:

```
FIND WATER
FIND ALL WATER
FIND ALL   THE SMALLISH SHALLOW   WATER
           [QUALIFICATION:]
           THAT IS [=]
                SOMEWHAT NEAR TO
                (OR   ( A GULLY
                       [A] FAIRLY SMALL MOUNTAIN ))
```

The presence of the verb FIND in the sentence causes a Locate program to be started. Since it requires an argument (a noun, or noun phrase) WATER becomes the argument of Locate. ALL is determined to be a qualifier from the set (none, some, all) and the Locate program uses this as a flag that all relevant instances found in the search are to be collected into the answer set. (If the term were SOME instead of ALL then a random selection of the answer set would be made and returned as the answer.)

The phrase THE SMALLISH SHALLOW (WATER) is used as a qualifier of WATER. A database search of all instances of water is made and the fuzzy qualifier SHALLOW is applied to all of them. SHALLOW is a continuous function which returns a grade of membership value which corresponds to an input depth. All of those water depths instances in the database whose depths is midde-SHALLOW or less (deep) are kept as the answer (set).

SMALLISH is a continuous function which returns a grade of membership value which corresponds to an input area or volume, depending on the context.. Of all the instances in the (SHALLOW) answer set those which further are middle-SMALL or less are retained as the answer set. SMALLISH will likely eliminate any ocean water or large lake-water from the answer set even though they are adequately SHALLOW.

The remaining answer set is reduced further by the qualifications specified as

[QUALIFICATION:]

THAT IS [=]  
SOMEWHAT NEAR TO  
(OR ( A GULLY  
[A] FAIRLY SMALL MOUNTAIN ))

which means that exists such that the stated qualification is met. The qualification is, in effect, SOMEWHAT NEAR A OR B, where A is GULLY and B is MOUNTAIN, and A and B have their qualifications (A and FAIRLY SMALL), respectively.

If no objects GULLY are found in the database then the system has to examine the geo-spatial database to detect / locate them based on a GULLY process. If there is no defined GULLY process then the system can ask the user questions, using geo-spatial (primitive) knowledge. The user might enter something like “A gully is a depression, like a very big rut, but may be as big as a very small valley.” The system knows what a depression is and a valley as well. If the system doesnt already know what a rut is it can determine (from the sentence (..like..)) that a rut is a variation of a depression, and through semantic inheritance (from depression) associate some property list elements (of depression) with rut.

As explained previously in this paper SMALL and FAIRLY are applied to the initial set of mountain instances and a set of answer instances (of mountain) results.

The A in A GULLEY resolves into “Some” GULLEY, which chooses one GULLEY from the answer set randomly and which is also SOMEWHAT NEAR... THE SMALLISH SHALLOW WATER instances set.

Next is discussion of the procedural attachment associated with the DAX ontology.

An example of a boolean determination of AtRight, is shown next, in Java.

```
public boolean AtRight( int x1, int x2 )
{
    if (x1 < x2 )
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Each of the RDF predicates in DAXSVG has a program member written in Java. These programs , like the one above, provide required actions for the DAXSVG ontology. The Java program above is purposely simplified for illustrative purposes. The nature of the simplification is this, the above program provides a boolean or true / false dichotomization of there being a “to the right” spatial relationship existing at the time of testing between two (SVG) objects whose (SVG) x-axis origin

values are  $x_1$  and  $x_2$ . The DAXSVG, as can be seen by reading the comments for the predicates in it, is used not as a boolean type system of true / false but rather a continuous system implementing shades of grey. A boolean function such as the one shown above can only detect “to the right” being true or false, all or nothing. Code used in association with the DAXSVG predicates computes a degree or how much the spatial (or other) predicate is true. The predicates themselves in the DAXSVG are conceptual identifiers they do not detect or compute the shade of grey themselves. Each DAXSVG conceptual identifier has associated with it one or more Java code modules. Together the DAXSVG predicates and the associated code modules comprise a kind of (collection of) ontological-object. Objects in computing have both a data aspect and an executable aspect (such as methods).

```
public real AtRight( int x1, int x2 )
{
    if (x1 < x2 )
    {
        return (near(abs(x2 - x1), contextx));
    }
    else
    {
        return 0;
    }
}
```

The fuzzy function  $f(x)$  calculates a sigmoidal curve, with  $k_1$  being the center of the curve, and  $k_2$  the slope of the curve. The concepts of Near-Far, for example, may be represented using this function. Far is computed by  $f(x)$ , and Near is computed as its complementary function  $f'(x) = 1 - f$ . Here  $f(x) = .5 + 1/\text{PI} \arctan ((x - k_1)/k_2)$ . Therefore (the fuzzy function) "near" in the above (fuzzy) AtRight function is calculated by  $\text{near} = 1 - (.5 + 1/\text{PI} \arctan ((x - k_1)/k_2))$ . Abs is absolute value calculation and contextx is the length of the reference x axis. The defined x axis SVG canvas value is used as the Context "contextx".

What the fuzzy version of the AtRight Java program calculates is : given a Cartesian plane where the SVG origin for it is at the upper left corner, then if the magnitude of the x coordinate of the item ( $x_2$ ) which might be to the right (of item  $x_1$ ) is greater than the magnitude of the other item ( $x_1$ ) then the function is to return the result of the computation shown where "true" occurs in the boolean version of AtRight. Otherwise return value of zero. The computation at "true" computes the distance between the two objects and uses that distance in the fuzzy calculation "near()", with Context = contextx. Near ( ) calculates a grade of membership (that "shade of gray" value, ranging between 0:1) which indicates a contextually based "how much" (to the right) value, which the AtRight function then returns. If, say,  $x_1$  is 20, and  $x_2$  is 30, then  $\text{AtRight}(x_1, x_2)$  returns a value such as 0.7. Zadeh has shown that a "linguistic variable" can be computed to "anglicize" the value "0.7" into a term such as “quite near” (to the right), or "to the right and quite close". Notice that these terms are real valued and not boolean.

## SVG means of capturing metaphorical and analogical knowledge

This is an SVG animation which the author wrote to demonstrate an XML (SVG) means of capturing metaphorical and analogical knowledge and providing visual / diagrams to represent such. The complete code for this SVG animation is at [open-meta.com](http://open-meta.com), called anim01.svg.

```

<?xml-stylesheet type="text/xsl" href="anim01.xsl"?>

<svg width="16cm" height="14cm" viewBox="0 0 255 201">
<desc>Copyright 2001 - 2005 David Dodds Example anim01a - demonstrate deBono
diagram SVG animation with Lakoff spatial metaphor</desc>
  <rect x="1" y="1" width="253" height="199"
    fill="black" stroke="blue" stroke-width="7" />

  <text id="uplabel" x="230" y="20"
    style="font-family:Verdana; font-size:12.333; fill:blue">
    UP
  </text>

  <text id="downlabel" x="200" y="180"
    style="font-family:Verdana; font-size:12.333; fill:blue">
    DOWN
  </text>

  <text id="goallabel" x="110" y="42"
    style="font-family:Verdana; font-size:12.333; fill:blue">
    GOAL
  </text>

  <g id="leftfunnel" side">
  <path d="M 99 180 L 99 57"
    style="fill:none; stroke:green; stroke-width:10"/>
  </g>

  <g id="rightfunnel" side">
  <path d="M 153 57 L 153 180 "
    style="fill:none; stroke:green; stroke-width:10"/>
  </g>

  <text id="Hide" x="15" y="190"
    style="font-family:Verdana; fill:orange; font-size:4; " >
  HIDE Scrolling Text
  </text>

  <text id="Explain" x="15" y="180"
  style="font-family:Verdana; fill:orange; font-size:4;"
  >
  Explanation of animation
  </text>

  <rect id="arrowstreamer" x="110" width="3" height="20" >
  <animate attributeName="y" attributeType="XML"
    begin="0s" dur="5s" fill="freeze" from="180" to="55" />
  <animate attributeName="height" attributeType="XML"

```

```

        begin="0s" dur="5s" fill="freeze" from="20" to="143" />
    <animateColor attributeName="fill" attributeType="CSS"
        from="rgb(0,0,255)" to="rgb(110,0,0)"
        begin="0s" dur="5s" fill="freeze" />
</rect>
<rect id="particlestreamer" x="120" width="3" height="3" >
    <animate attributeName="y" attributeType="XML"
        begin="2s" dur="5s" fill="freeze" from="170" to="55" />
    <animateColor attributeName="fill" attributeType="CSS"
        from="rgb(0,0,255)" to="rgb(110,0,0)"
        begin="2s" dur="5s" fill="freeze" />
</rect>

    <rect id="misdirectedbehaviourstreamer" x="147" y="190" width="5"
height="5"
>
    <animate attributeName="y" attributeType="XML"
        begin="3s" dur="5s" fill="freeze" from="190" to="130" />
    <animate attributeName="x" attributeType="XML"
        begin="3s" dur="5s" fill="freeze" from="147" to="230" />
    <animateColor attributeName="fill" attributeType="CSS"
        from="rgb(0,0,255)" to="rgb(128,0,0)"
        begin="3s" dur="5s" fill="freeze" />
</rect>

    <rect id="obstructor" class="hitBox" x="40" y="49" width="50" height="5"
>
    <animate attributeName="x" attributeType="XML"
        begin="3s" dur="5s" fill="freeze" from="40" to="100" />
    <animateColor attributeName="fill" attributeType="CSS"
        from="rgb(10,0,0)" to="rgb(255,0,0)"
        begin="3s" dur="3s" fill="freeze" />
</rect>

    <text id="obstructorlabel" x="120" y="53"
        style="font-family:Verdana; font-size:4; fill:black">
    <animateColor attributeName="fill" attributeType="CSS"
        from="rgb(0,0,0)" to="rgb(255,255,255)"
        begin="7s" dur="2s" fill="freeze" />
    obstructor
</text>

    <text id="misdirectedbehaviourlabel" x="200" y="125"
        style="font-family:Verdana; font-size:4; fill:black">
    <animateColor attributeName="fill" attributeType="CSS"
        from="rgb(0,0,0)" to="rgb(255,255,255)"
        begin="9s" dur="3s" fill="freeze" />
    misdirectedbehaviour
</text>

```

</svg>

SVG can have embedded meta-data, it can also be parsed by an XML parser, XSLT can treat SVG via its “patterns” and transform it into something else XML. SVG can be used “metaphorically” or analogically to represent the process of cutting, such as cutting wire-link fence. That can in turn be rasterized so that a program can “know what to look for” in a tv image of cut material, such as fence.



### Very brief excerpt from University of Ottawa 'Context' ontology

See the paper “Dodds 2004” at open-meta.com for a detailed explanation as to how the context ontology is used to perform situated actions (rather than simply context-free actions). The situation is what is used to know which fuzzy calculations to perform and what their parameters should be (ie their context).

“<http://site.uottawa.ca/Context#>”

```
<daml_oil:ObjectProperty rdf:ID="thingRole">  
  <daml_oil:domain rdf:resource="http://site.uottawa.ca/Context#ThingRole"/>  
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#anyType"/>  
</daml_oil:ObjectProperty>
```

```
<daml_oil:Class rdf:ID="Action">  
  <rdfs:subClassOf rdf:resource="http://site.uottawa.ca/Context#Social"/>  
</daml_oil:Class>
```

```
<daml_oil:Class rdf:ID="ActivityPlace">  
  <rdfs:subClassOf rdf:resource="http://site.uottawa.ca/Context#Building"/>  
  <rdfs:subClassOf rdf:resource="http://site.uottawa.ca/Context#OutsidePlace"/>  
</daml_oil:Class>
```

```

<daml_oil:ObjectProperty rdf:ID="thingName">
  <daml_oil:domain rdf:resource="http://site.uottawa.ca/Context#ThingRole"/>
  <daml_oil:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#anyType"/>
</daml_oil:ObjectProperty>

```

```

<daml_oil:Class rdf:ID="ContextFeature">
</daml_oil:Class>

```

```

<daml_oil:Class rdf:ID="Conference">
  <rdfs:subClassOf rdf:resource="http://site.uottawa.ca/Context#Action"/>
</daml_oil:Class>

```

```

<daml_oil:Class rdf:ID="ThingDescription">
  <rdfs:subClassOf rdf:resource="http://site.uottawa.ca/Context#ContextFeature"/>
</daml_oil:Class>

```

Only a very small amount of the overall U of O 'Context' is shown here, the purpose here to give a peek at the appearance of a DAML-OIL-based ontology to get a sense for how they are the same and different from RDFS-based ontologies.

The notion of context used by David Dodds encompasses some of the U of O ontology but does not include other parts of it. If interested in the details of the usage / definition of Context in my works see the data-sets and papers at my site open-meta.com. Part and parcel with context I include the notion of situation-awareness, where situation is a particular case of context and detection of the occurrence of a particular variable activity is used as “awareness”.

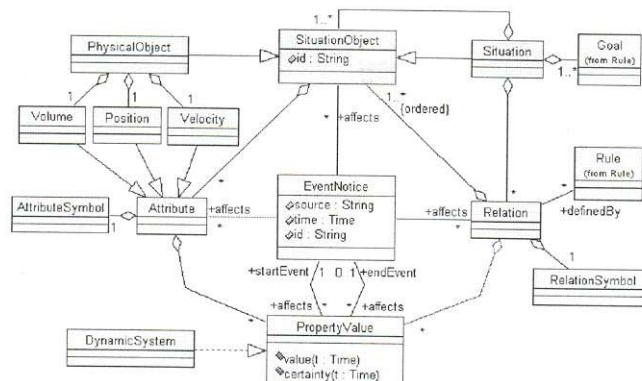


Figure 1. Core SAW Ontology

## References

- [1] David Dodds. "Fuzzy Logic Computer Implementation of Metaphor from Ordinary Language". 1981. AAAS Annual Meeting.
- [2] David Dodds. "Fuzziness in Knowledge-Based Robotics Systems". 1988. Fuzzy Sets and Systems 26: North-Holland 179-193.
- [3] David Dodds et al. "WROX Professional XML Meta Data". 2001. Wrox Press Inc; ASIN: 1861004516 600 pages.
- [4] David Dodds. "Scalable Vector Graphics 1.0" 2001. An Author of the language.
- [5] David Dodds. "Accessing SVG Content Linguistically and Conceptually". 2003. SVGOpen, Vancouver.
- [6] David Dodds. "Fuzziness in Approximate and Common-Sense Reasoning in Knowledge-Based Robotics Systems". The International Society for Optical Engineering, Space Station Automation III, SPIE Vol 851, 1987.
- [7] David Dodds. "Programming SVG Advanced Access Using Metadata and Fuzzy Sets". 2003. SVGOpen, Vancouver.
- [8] David Dodds. "Extending Representation Capability". Extreme Markup Languages 2004.
- [9] David Dodds. "Natural Language Processing and Diagrams". The 2004 International Conference on Machine Learning; Models, Technologies and Applications; which is a part of The 2004 International Multiconference in Computer Science and Computer Engineering. Las Vegas.
- [10] David Dodds. "Components of Meta-Programming, Computer Analogies and Metaphors". The 2005 International Conference on Programming Languages and Compilers, Las Vegas.
- [11] David Dodds. "Navigation of an Autonomous Mobile Intelligence Unit in the Unstructured Natural Environment." Second Workshop on Military Robotic Applications, Royal Military College of Canada and Defence & Civil Institute of Environmental Medicine, National Defence, 1989.
- [12] David Dodds. "Common Sense Planning Applied to Grasping and Manipulating", The International Society for Optical Engineering, Space Station Automation III, SPIE Vol 851, 1987.
- [13] David Dodds. "Use of Spatial Metaphor in World Model Representation, Spatial Planning and Robotic System Task Definition". Proceedings of Robotic Intelligence and Productivity Conference, Detroit, November 1983.
- [14]. Kurt Schmucker. "Fuzzy Sets, Natural Language Computations, and Risk Analysis". 1984. David Dodds is in bibliography.
- [15] Russell Suereth. "Developing Natural Language Interfaces", McGraw-Hill, C source code CD, ISBN 0-07-913017-8.
  
- [16] OntoSensor, Department of Electrical and Computer Engineering, University of Memphis, TN
  
- [17] GML Simple Feature

david\_dodds\_2001@yahoo.com

A bibliography and code listings are available for viewing, contact the author. A much more extensive / detailed version of this paper will be available , at **open-meta.com**.

## Listing F.2.2.2 schools.xml

```
<gml:name>School districts in the North Region.</gml:name>
<gml:boundedBy>
<gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coordinates>0,0</gml:coordinates>
<gml:coordinates>50,50</gml:coordinates>
</gml:Box>
</gml:boundedBy>

<gml:featureMember>
<SchoolDistrict>
<gml:name>District 28</gml:name>
<gml:boundedBy>
<gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coordinates>0,0</gml:coordinates>
<gml:coordinates>50,40</gml:coordinates>
</gml:Box>
</gml:boundedBy>
<schoolMember>
<School>
<gml:name>Alpha</gml:name>
<address>100 Cypress Ave.</address>
<gml:location>
<gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coordinates>20.0,5.0</gml:coordinates>
</gml:Point>
</gml:location>
</School>
</schoolMember>
<schoolMember>
<School>
<gml:name>Beta</gml:name>
<address>1673 Balsam St.</address>
<gml:location>
<gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coordinates>40.0,5.0</gml:coordinates>
</gml:Point>
</gml:location>
</School>
</schoolMember>
<gml:extentOf>
<gml:Polygon srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:outerBoundaryIs>
<gml:LinearRing>
<gml:coordinates>0,0</gml:coordinates>
<gml:coordinates>50,0</gml:coordinates>
<gml:coordinates>50,40</gml:coordinates>
```

```
<gml:coordinates>0,0</gml:coordinates>
</gml:LinearRing>
</gml:outerBoundaryIs>
</gml:Polygon>
</gml:extentOf>
</SchoolDistrict>
</gml:featureMember>
```

```
<gml:featureMember>
<SchoolDistrict>
<gml:name>District 32</gml:name>
<gml:boundedBy>
<gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coordinates>0,0</gml:coordinates>
<gml:coordinates>30,50</gml:coordinates>
</gml:Box>
</gml:boundedBy>
<schoolMember>
<School>
<gml:name>Gamma</gml:name>
<address>651 Sequoia Ave.</address>
<gml:location>
<gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coordinates>5.0,20.0</gml:coordinates>
</gml:Point>
</gml:location>
</School>
</schoolMember>
<schoolMember>
<College>
<gml:name>Delta</gml:name>
<address>260 University Blvd.</address>
<gml:pointProperty>
<gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:coordinates>5.0,40.0</gml:coordinates>
</gml:Point>
</gml:pointProperty>
</College>
</schoolMember>
<schoolMember xlink:type="simple" xlink:title="Epsilon High School"
xlink:href="http://www.state.gov/schools/cgibin/
wfs?schoolID=hs736" gml:remoteSchema="schools.xsd#xpointer(//complexType
[@name='SchoolType'])"/>
<gml:extentOf>
<gml:Polygon srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
<gml:outerBoundaryIs>
<gml:LinearRing>
<gml:coordinates>0,0</gml:coordinates>
```

```

<gml:coordinates>40,50</gml:coordinates>
<gml:coordinates>50,50</gml:coordinates>
<gml:coordinates>0,0</gml:coordinates>
</gml:LinearRing>
</gml:outerBoundaryIs>
</gml:Polygon>
</gml:extentOf>
</SchoolDistrict>
</gml:featureMember>

```

The area of North Region=  $\text{area}(0,0,50,50)$ .

The area of District 28=  $\text{area}(0,0,50,40)$ .

The area of District 32=  $\text{area}(0,0,30,50)$ .

North Region contains District 28 and District 32.

District 28 is somewhat smaller than North Region  
 [because  $\text{maxdistx} = \text{comparison basebox size (North Region=)} 0,0,50,50$ ]

Gamma School (at 5.0,20.0) is quite near Delta College (at 5.0,40.0)  
 [because  $\text{maxdistx} = \text{comparison basebox size (District 32=)} 0,0,30,50$ ]

Beta school is somewhat to the right of Alpha school.  
 [because  $\text{maxdistx} = \text{comparison basebox size (District 28=)} 0,0,50,40$ ]

```

<rdf:Property ID="Near">
<rdfs:comment>has a degree of nearness (by value).  $g_1(x)$ </rdfs:comment>
<rdfs:range rdf:resource="#www.open-meta.com/2001/IsNear" />
<rdfs:domain rdf:resource="#SvgEntity" />
</rdf:Property>

```

$g_1(\text{centroidsdistance})$

$\text{centroidsdistance} = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$   
 $\text{centroidsdistance} = \sqrt{(5.0-5.0)^2+(40.0-20.0)^2}$

$\text{Near} = g_1 = 1 - (1/2 + 1/\text{PI} * \arctan(\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}) - k(1)/k(2))$   
 $\text{Near} = g_1 = 1 - (1/2 + 1/\text{PI} * \arctan(\sqrt{(5.0-5.0)^2+(40.0-20.0)^2}) - k(1)/k(2))$

$(\text{Near}(5.0,20.0,5.0,40.0))$