January 28, 2013

Knowledge-Based Environmental Context Modeling

Paul Pukite, Steve Bankes, Dan Challou: BAE Systems
Thomas Huang: JPL

Abstract: This paper describes a semantic web architecture based on patterns and logical archetypal bulding-blocks well suited for comprehensive environmental modeling framework. The patterns span a range of features that cover specific land, atmospheric and aquatic domains intended for terrestrial and amphibious vehicles. The modeling engine contained within the server relied on knowledge-based inferencing capable of supporting formal terminology (through the SWEET ontology and a domain specific language) and levels of abstraction via integrated reasoning modules.

Contents

Introduction

The design of ruggedly complex systems such as advanced ground vehicles requires knowledge of the environmental contexts that occur during operational deployment. For example, the nominal and extreme terrain and weather conditions have an impact on the choices made in the vehicle design. A test vehicle has to endure and even thrive in extreme conditions to earn the title of a ruggedized design.

If you want to know whether a vehicle can travel over hilly terrain, it's a good idea to understand the extent of the hilliness in the region, and characterize that in terms of probability of elevation changes.

If you want to estimate how a vehicle will respond to a rocky road, it makes sense to characterize the bumpiness and fine relief structure and formulate that in terms of a suspension and chassis simulation model.

If you want to predict how a vehicle will respond to windy conditions, it is useful to have estimates and likelihood of the extreme conditions that may arise for that climate.

Figure 1: *Examples of rationale for characterization and context modeling. Our concentration is on the invariant aspects of the environment that exist independent of the subject's role. These are considered non-compliant one-way relationships, as a vehicle can't impact certain exogenous properties.*

Based on these considerations, we need a comprehensive approach for aggregating model knowledge. The approach that we outline below extends from research describing the elements necessary to create workflow architectures for vehicle design[1]. In particular, the building blocks that originated from design-centric semantic web features together with knowledge-based reasoning were deemed general enough to find applicability to an environmental modeling framework (see **Figure 2**).

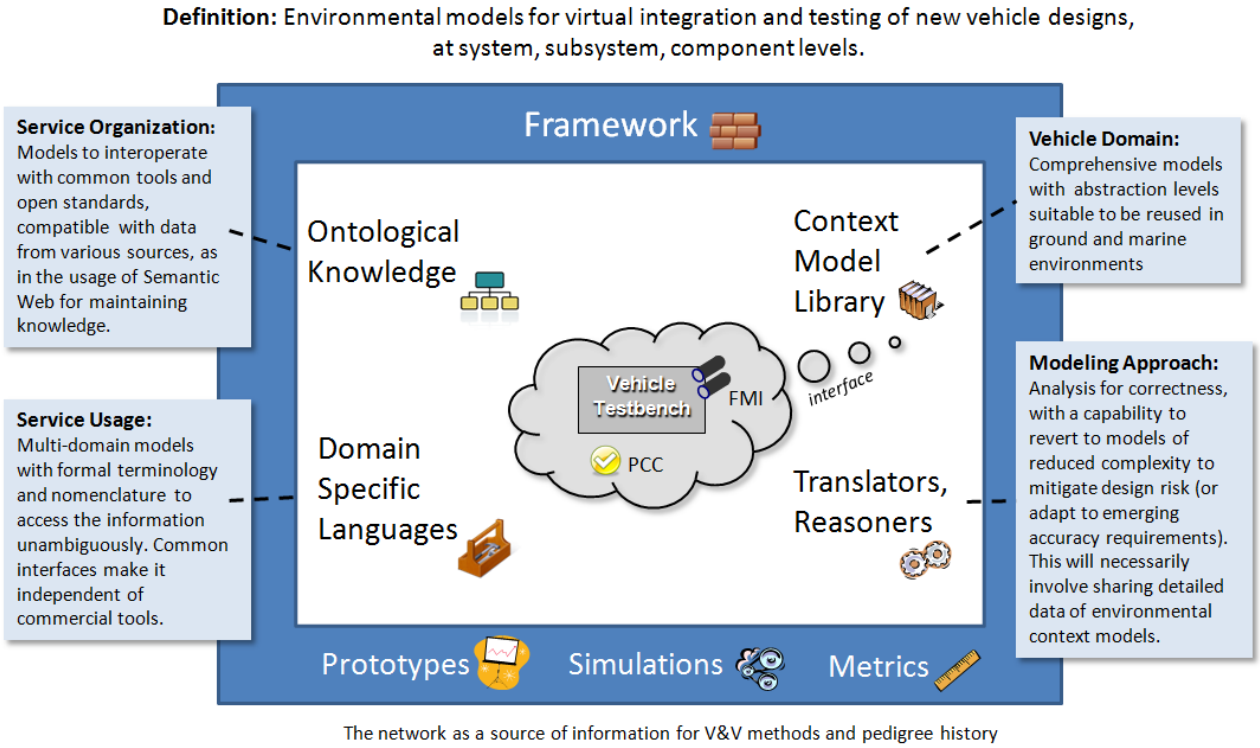


Figure 2*: View of context modeling framework with emphasis on vehicle design. For virtual simulation, the innermost vehicle test-bench needs to interface to the environmental context model.*

Starting from a general design and workflow approach outlined elsewhere[1] , we applied comparable knowledge-based patterns to environmental modeling (see **Figure 3** below). At the modeling level, we

consider abstractions, organization, search, and logical reasoning to improve the convenience and usability across the suite of models.

Since the models are quantitative and often statistical in nature, we employ a concise descriptive language to declaratively specify their behavior. Simulation abstractions allow us to transition between purely data-driven and probabilistic views of models. The convenience of automated searching and reasoning benefited from the semantic-based organization of the underlying knowledge-base.

	Examples	Benefits
Domain Specific Language ∑	Statistical, complex, and matrix math.	Concise, symbolic, declarative, with formal representations
Abstraction Levels 	Local/global spatial levels, probability distributions	Comprehensive, detailed, and aggregated views of environmental contexts
Organization 	Semantic web ontologies such as SWEET	Classification, maintenance, services, library curation
Searching 	Linked data elements, keyword and classification	Models discovered through requirements linkage, etc
Reasoning 	Workflow and parametric modeling	Guided model generation and usage.

***Figure 3**: Knowledge-based modeling patterns*

The objective is to present the environmental view of the system as a dynamic context server (DCS). This provides a flexible framework for handling interactive applications such as guided workflows, artifact viewing, and reusable web services.

The DCS was organized with a breadth-first view to make sure that the essential semantic, ontological, logical, and mathematical capabilities were at least tangentially covered. We start with a set of categorical domain features which cover land, atmosphere, and aquatic environments, and then apply a set of facets to these categories to allow the user various levels of access to the knowledge. The orthogonal axes are illustrated in **Figure 4**.

The basis for the modeling is described in a set of foundation papers [2][3][4] which applied stochastic patterns to the empirical observations, leading to a set of concise formulations. The semantic organization provided a means to manage the growing array of patterns. Much as a scientific library provides organization among the subject domains, the semantic layers afforded a similar discipline to the model hierarchy. In practical terms, we applied the concept of archetypes as patterns of use[1] and organize our knowledgebase at the semantic level to take advantage of the reuse and commonality available by suitable classification.

***Figure 4**: The general semantic organization corresponds to modeling domain features arranged against user facets. The features provide semantic keying for various levels of search. In terms of the SWEET ontology, the features correspond to environmental categories, while the facets are human-centric.*

Building Blocks

The foundation of the context server is built on a knowledgebase based on triple-store technology and managed by a declarative first-order logic language. The language chosen, Prolog[5], bridged the gap between the (1) low-level structures of the underlying triple-store database and the (2) declarative structure of an abstracted user interface[6]. The rules and inferencing capabilities of the language provided sufficient expressivity to allow the context server to function as a flexible semantic web-server[7].

In the following overview, we will cover the following building blocks at least briefly:

- Integration with triple-store (RDF, OWL, SWEET)
- Knowledge base and logic formulation
- Entailment – creating rules that abstractly appear as triple store
- Search and query – Semantic Web servicing – integration of queries as services
- Processing data formats such as XML and JSON
- Ontology graphing
- Generators for code production and data
- Math and complex math
- Array and list processing
- Symbolic and dimensional unit processing
- Integration with statistics framework such as R

- Artifact generation such as plots for PDF and PSD

Triple Store: The triple-store data format is the backbone of the semantic web. A triple consists of a *(Subject, Predicate, Object)* tuple which can be linked to other triples in seemingly infinite combinations, leading to significant flexibility and robustness in a database.

The structure of a triple, cast in a conventional *resource description format* (RDF)[8] or by the XML-free Turtle format, essentially reduces to mapping against what is referred to as a Prolog functor. So the following query maps directly to a RDF triple-store:

```
rdf(Subject, Predicate, Object)
```

From a programmatic level, that is all there is to interfacing Prolog to a triple-store. The semantics turns out to be equivalent to that of a SPARQL query[9], where the binding of the arguments to either variables or ground terms determines what will be returned from the query. Several useful guides are available which describe the essential match between the Prolog binding mechanics and that governing SPARQL queries and description logic written in straight RDF. Further work has been done in mapping to higher level descriptions built on top of RDF, such as in the *web ontology language* known as OWL[8].

We take advantage of a ready-made OWL classification system set up for environmental sciences modeling by using the terminology defined by the OWL-based *Semantic Web for Earth and Environmental Terminology* (SWEET) ontology[10][11]. The key to using ontologies such as SWEET is to create triple-store instance data which subclasses or keys off the classification system defined by the SWEET environmental hierarchies (see **Figure 5**).

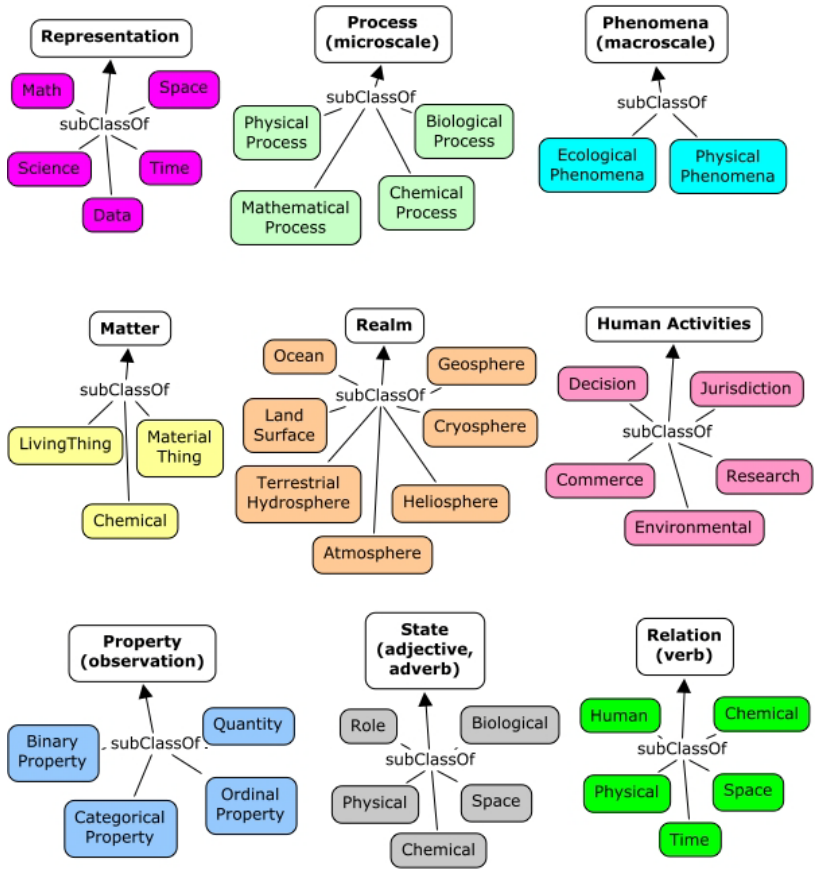


Figure 5: *SWEET hierarchies and environmental classifications. The enclosed terminology is sufficient to define the majority of the context modeling knowledge.*

We apply an additional abstraction, which is quite common in practical semantic web applications, which is to define a namespace layer between the instance data and that defined by SWEET. In Figure 6 below, we show several DCS-defined sub-classed resources that start with the letter 'a', and associate that with the SWEET terminology class that it best fits into.

Context Server Resource	SWEET (rdfs:subClassOf)
ent:atm	reprSciUnits:Unit
ent:atomicUnit	matrParticle:Atom
ent:averageAlbedo	propFraction:Albedo
ent:averageSolarInsolation	propEnergyFlux:Insolation
ent:avogadrosNumber	propQuantity:PhysicalConstant

Figure 6: First several instances in graph ent = http://entroplet.com/terms# sorted by label

Suffice to say that the SWEET terminology supplies very complete coverage to the environmental modeling features required in the context server, and we have updated SWEET in the few areas that were deficient in representation power.

Knowledgebase: The context server knowledgebase is a composite of all the information stored in the triple-store database and the rules and facts stored in the Prolog repository. The most elementary form of a rule is to express the logic as some form of constraint, relationship, query, etc.

Rules can be used to define a specific closure or find a specific closure in the underlying triple-store database (see Figure 7)

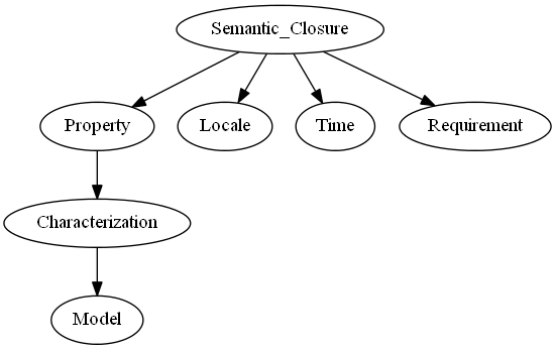


Figure 7: Linked triple-store features from a top-level rule define a semantic closure.

Entailment: A powerful feature of the correspondence between the declarative semantics of Prolog and a triple-store query is that we can create rules that abstractly function as triple stores (or that can extend the triple store). Per the semantic definition, *entailment* is the principle that the truth of one statement ensures the truth of a second statement. For example, we can extend an RDF query[12] that returns a resource with a real-valued component, so that we do not have to perform further parsing to convert the string to a float.

```
    rdfR(Subject, Predicate, RealObject)
```

Search and Query: We use a version of Prolog that contains a complete web servicing library (called Cliopatria[6]). This allows searches and queries to be composed as web requests. For example, the following URL request searches for resources that fit within a specific SWEET category (in this case a wind category):

```
http://localhost:3020/context_search/list_cats?name=phenAtmoWind:Wind
```

This works on two levels, the first stage of processing decomposes the URL into an internal format via REST (representational state transfer) semantics, and the second level, which searches the triple-store database (or Prolog knowledgebase) for matches, and then presents the results to the client.

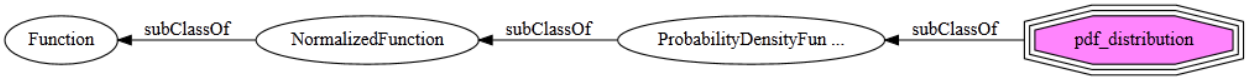
As another example, this query requests a model of ocean clouds (see Annex B)

```
http://localhost:3020/context_model/apply?loglog=false&model=Ocean_clouds
```

Semantic web servicing and specifically the integration of queries as services allow us to build up the capabilities of the context server. In that sense, we can use (1) queries as model services, (2) queries to generate algorithms and code, (3) queries to generate artifacts and metrics, and so on.

Data Formats: Processing data formats such as XML and JSON is important with respect to reading from externally stored knowledge and to providing data to the user. Prolog is essentially a list-processing language and much of the XML and JSON translation is conveniently available through the built-in libraries.

Ontology Graphing: Also built-in as a library is graphical representation for the knowledgebase and its semantic hierarchy. For example, the image below represents the sub-classing of a context modeling class, `ent:pdf_distribution`, with respect to the SWEET terminology.



Code Generation: Many environmental models are stored as mathematical representations, intended to express a model's behavior as closely to its original abstract form as possible. If this format is unambiguous and thus amenable to straight-forward parsing, the representation can easily be transformed into any software language that can also represent the mathematical expressions.

In certain cases, we have chosen to represent models in a symbolic math format. Here, we can create a suite of translators for conversion from that format to an arbitrary language with the necessary mathematical expressiveness. We have shown below an example for a temperature model, where **T** is the calendar time and **S** returns the symbolic temperature expression.

```
%    Generate a symbolic algebraic expression
calculate_symbolic_temperature(Site, T, S) :-
    rdfS(U, ent:name, Site),
    rdfR(U, ent:t0, T0),
    rdfR(U, ent:ty, Ty),
    rdfR(U, ent:dT, DT),
    rdfR(U, ent:td, Td),
    rdfR(U, ent:a, A),
    rdfR(U, ent:b, B),
    rdfR(U, ent:c, C),
    PI is pi,
```

```
    S = T0+Ty*sin(2*PI/365*T+A)+(DT*sin(2*PI/365*T+B)+(Td-DT))*sin(2*PI*T+C).
```

The expression **S** can be evaluated directly as below, where the term " is " directs the interpreter to evaluate the expression

```
%    Generate a numeric from symbolic
calculate_numeric_temperature(Site, T, Temperature) :-
    calculate_symbolic_temperature(Site, T, Symbolic_Temperature),
    Temperature is Symbolic_Temperature.
```

Since **S** is a symbolic representation, it can straightforwardly be translated to C-code, Python, etc. We will demonstrate this later.

Math and Complex Numbers: To minimize the potential complexity of the mathematical representations, we can take advantage of certain idioms that are heavily used and maintain those as domain specific representations. In that sense, we can create a domain specific language to handle the essential and common aspects of math-based models.

Standard Prolog is expressive enough to handle much of the math load as shown with the code production example described previously, yet we find that a mini-language is useful to handle array processing and complex number math.

As an example of the potential conciseness of representation, the complete declarative representation of a recursive Fast Fourier Transform (FFT) is the following (the *product_and_sum* and *evens_and_odds* are helper functions, and *w* is a lookup table).

```
%    Fast Fourier Transform
fft(1, Ft, Ft) :- !.
fft(N, F, Ft) :-
    N > 1,
    N2 is N // 2,
    evens_and_odds(F, E, O),
    fft(N2, E, Et),
    fft(N2, O, Ot),
    w(1, W1),
    w(2, W2),
    w(N, Wn),
    product_and_sum(Et, Ot, W2, Wn, Gt, []),
    product_and_sum(Et, Ot, W1, Wn, Ft, Gt).
```

The function *product_and_sum* illustrates how we have abstracted the expression evaluator " is " (described in the previous code production example) to " isx " which now evaluates expressions involving complex numbers. A complex number is represented as $a + i\,b$, where i represents the imaginary number $\sqrt{-1}$.

```
product_and_sum([], [], _, _, Ft, Ft).
product_and_sum([E| Et], [O| Ot], Wk, Wn, [F| Ft], Fu) :-
    Temp isx O * Wk,
    F isx E + Temp,
    Wk1 isx Wk * Wn,
    product_and_sum(Et, Ot, Wk1, Wn, Ft, Fu).
```

This complex number formulation is used for calculating terrain profile power spectral densities (PSD) based on input data with array sizes of several hundred thousand elements.

Array and List Processing: Concise list array processing is useful to represent and generate the data elements described in most of our stochastic environmental models. Complex number list processing is used for calculating PSD's as shown in the complex math FFT example, while lists comprised of real numbers are used for calculating probability density functions (PDF).

As a domain specific representation we chose to apply list and array operators with *infix* notation wherever possible. The processing function then reads as *Y results from applying against X*.

Y ***mapdot*** *X*
Map Dot is a dot product of individual elements, retaining the list structure.

For this specific case we take an example from the Matlab world where the operator " **.*** " is defined to multiply a scalar by a vector, so that the following relation holds:

```
[20,40] mapdot  2 .* [10, 20]
```

This expresses the invariant that 20 is the doubling of the first element in the list, and 40 is the doubling of the second element of the list.

For further expressiveness, the term " **~>**" is used to indicate a map apply to a list:

```
[2.0, 4.0] mapdot 2 * sqrt ~> [1.0, 4.0]
```

A number of specialty array and list processing functions follow this infix idiom:

Y ***range*** *[From, To]*
Range of numbers into a list

Y ***dot*** *A*B*
Dot product of two arrays or lists of the same length

Y ***convolve*** *A*B*
Convolve (convolution operator) list with another list.

Y ***correlate*** *A*B*
Pair correlates one list with another list.

Y ***derivative*** *A/B*
Take the derivative of one list with respect to another list.

Y ***integrate*** *A*B*
Take the integral of one list with respect to another delta list.

Y ***difference*** *A-B*
Take the difference of one list with respect to another list.

Y ***tuple*** *A+B*
Create a tuple list from two lists.

Y ***unbias*** *X*
Remove the mean from a list of values, so the sum is zero.

Y ***normalize*** *X*
Normalize a list of values, so the sum is unity.

Y ***pdf*** *X*
Create a PDF from a list of values.

Y ***zshift*** *A-B*
Do a DSP z-shift from a list.

Y ***window*** *X/A*
Y ***window*** *X*A*
Do a lag-window or centered window on a list.

Y ***shrink*** *A/B*
Shrink a list to match the second.

Y ***expand*** *A+B*
Expand a list to match the second.

Y ***cat*** *X*
Flatten a list.

Y ***offset*** *X*
Offset a list by N elements.

Y ***ordinal*** *X*
Create an ordinal (counting) list from a list.

Y ***split*** *X*
Split pairs of values into two sequential lists.

As an example of the sophistication of use, these four lines generate a number line, a sine and cosine valued list and then a list of (*X*, *Y*, *Z*)-tuples suitable for plotting as a graph:

```
X range [1,50]/0.1,
Y mapdot sin ~> X,
Z mapdot cos ~> X,
Graph tuple X + Y + Z
```

Or consider this case of concisely generating a cumulative density function against an *X* variate of a given range, and then producing a probability density function from that result

```
X range [0,100]/0.1,
Y mapdot exp(10) ~> X,  % create CDF of exponential with damping factor 10
Z pdf Y
```

The final objective was to be able to abstractly represent fairly sophisticated list computations in as concise a form as possible. As is the case with many of these domain specific idioms, the result is that a user can scan the code and be able to quickly determine the transformation that is being applied.

Statistical Integration: A language such as Prolog is well-suited for (1) general purpose logic programming, (2) creation of domain specific languages (as shown with the complex and list processing DSLs described above) and (3) application development such as is required for a semantic web server.

Where it falls short is in the area of special-purpose computing such as in statistical and scientific computation, which may require extensive matrix handling and comprehensive math libraries. The open-source statistical computational tool known as **R** is well-suited to corners of the statistical and

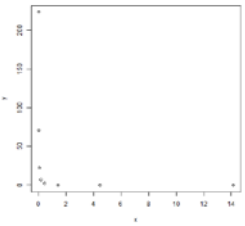
mathematical world that basic Prolog does not have a foothold in[1] .

Fortunately, the version of Prolog used on this project has an API to **R** and features a domain specific language abstraction defined specifically for data processing. Similar to what we use for list processing, an equivalent list processing interface is defined to **R**.

In particular, the infix operator " <- " is used to generate an input/output transfer from Prolog to **R** (and the reverse). So in the following, we generate a list profile called *Input*, but then request that **R** perform a *BesselK* function lookup on that list and attach it to an **R** state vector called *y*.

```
Input mapdot sqrt ~> 2.*[0.00001,0.0001,0.001,0.01,0.1,1.0,10.0,100.0],
y <- besselK(Input,1),
Y <- y,
Output mapdot Input * Y
```

The rationale for doing this is that Prolog does not have a library of transcendental functions such as the set of modified Bessel functions (which are used in PDF models), yet **R** does contain a comprehensive library, largely due to its statistics underpinnings. The fact that the list processing idioms that we have in place match well with the list I/O of the **R** library makes this a good combination for corner-case modeling needs, and where more sophisticated graphing is needed (see the contour plots in Section X for an example). The combination of Prolog with **R** is comparable to the capabilities of Matlab.



The statistical leverage that **R** provides to the semantic web server is crucial for further enhancements to context modeling.

HTML Development: The semantic web library includes convenient mechanisms for developing HTML code, well suited for making sophisticated interactive web pages. The declarative style of programming used in Prolog allows a very similar but much more concise and powerful representation than does the XSLT (Extensible Stylesheet Language Transformations) language[13] used in conventional web page development.

The library uses what is called definite clause grammars (DCG) to generate valid HTML[14], with the code remaining very readable. A DCG is much like a XSLT fragment, allowing for recursion and pattern matching, but presented in a much more compact and concise style.

```
dispatch(wind) -->
    html([h1('Wind models'),
          \g(search,
             a([href('/context_model/navigate?characteristics=windSpeed'),
                target(target_iframe)], 'Wind PDF models')),
          \(ref_search('phenAtmoWind:Wind', 'Wind references')),
          \g(browse,
             a([href('/context_browse/navigate?term=atmospheric')],
               'Browse atmospheric characteristics')
            )
         ]
        ).
```

This HTML, when rendered, appears as the following style-sheet-based representation (see **Figure 8**):

[1] See http://blog.revolutionanalytics.com/2012/07/a-big-list-of-the-things-r-can-do.html for a list of R application areas.: Basic math and statistics, probability distributions, big data analytics, optimization and mathematical programming, signal processing, simulation and random number generation, statistical modeling, statistical tests, static and dynamic graphics.

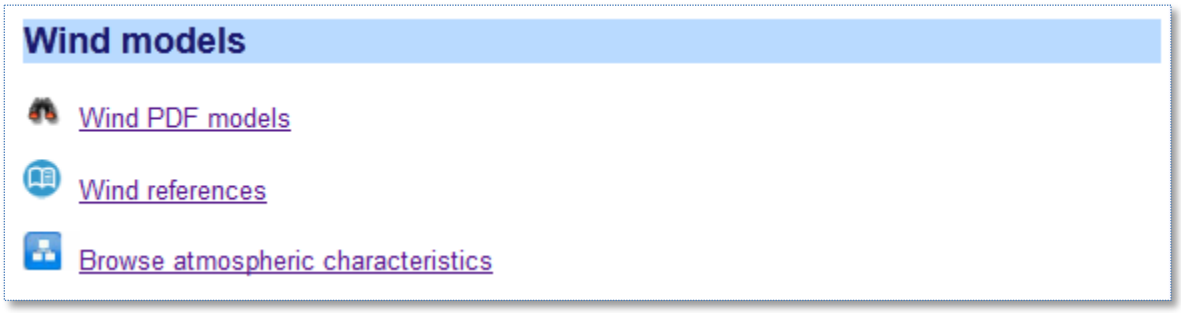


Figure 8 : *Rendered HTM from a Prolog DCG rule.*

The terms leading with the character '\' call out user-defined DCG functions which will create in-place snippets of HTML code. In practice, by generating the majority of the HTML code using a mix of logic and meta-logic, we employ what amounts to a hybrid of a PHP hypertext preprocessor[15] (executed on the server side) and the declarative nature of XSLT (but without the excessively verbose syntax of the latter).

In general and for most HTML coding we use this idiom, and only for certain applications, such as in form input look-ahead and interactive graphing, do we drop into JavaScript.

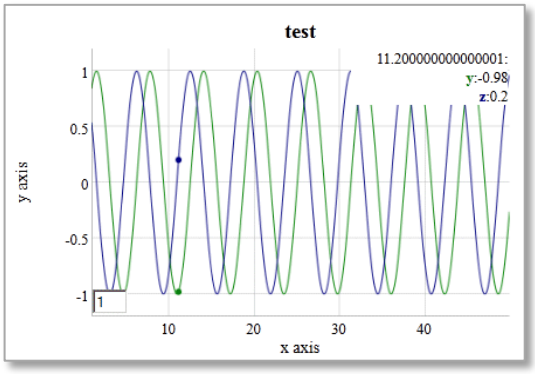
Plot Artifacts: Generating artifacts such as plots of PDF and PSD models is facilitated by the use of open-source JavaScript libraries that feature dynamic interaction.

From either models or data, we use dynamic graphs to construct PDF, CDF, and PSD profiles, along with expected value plots of annual, diurnal, etc measures and various growth curves.

The graphing interactive features include linear or log scales, adjustable noise filtering, the inspection of data points, and panning and scaling.

As an example, we extend the list processing example described earlier to plot a dynamic graph, which consists of two trigonometric functions separated by a phase shift:

```
dygraph_test(_) :-
   X range [1.0,50.0]/0.1,
   Y mapdot sin ~> X,
   Z mapdot cos ~> X,
   Graph tuple X + Y + Z,
   reply_html_page([title('chart'), \(con_text:style_cliopatria)],
                   [\(context_graphing:dygraph_native(
                                          false,
                                          ['x', 'y', 'z'],
                                          'x axis',
                                          'y axis',
                                          'test',
                                          Graph))]).
```



Random Number Generation: A module for random number generation is an important building block for Monte Carlo simulations. A uniform random variate generator can be used in conjunction with invertible PDF profiles to create sampled data for exponential, Bessel, Rayleigh, fat-tail, and several other distributions.

Terrain profile generators also rely on a random number generator to create random walk profiles of Markov and semi-Markov character. For aquatic wave generation, random superposition of sine waves is effective in modeling the empirical characteristics observed.

Dimensional Units Checking: The framework uses an embedded Prolog reasoner to parse and pattern match if dimensional unit conversion is required on some measure. The unit symbology is retained as lexical terms with both proportional scaling or inverse scaling allowed, depending on whether a "*" or "/" appears in the unit dimensionality string. A database of root units and at least one relation to another unit of the same measure is kept in the local knowledgebase so any combination can potentially be parsed and evaluated.

As an example of a relatively complex unit string, we take the Stefan-Boltzmann constant specified in terms of SI units, `5.67e-8*j/s/m^2/k^4`, and can use the unit checking module to convert this value to a hybrid mix of English units:

```
?- context_units:convert(5.67e-8*j/s/m^2/k^4, SB*btu/hr/ft^2/r^4, SB).
SB = 1.7121822360279162e-9.
```

Where j=joules, s=seconds, m=meters, k=kelvin on the original, and btu=BritishThermalUnits, hr=hour, ft=foot, and r=Rankine on the desired conversion.

This feature is valuable in reducing the amount of translation code that is required, as the number of combinations is nearly unlimited but the parser is able to perform the reduction symbolically.

Diagramming Relationships: Similar to the library for **R** interoperability, the semantic web infrastructure also integrates to the directed-graph library known as *Graphviz*. The *Graphviz* rendering generates SVG (scalable vector graphics) which is compatible with most HTML browsers.

This capability allows us to visualize closure of rules as a directed graph, and look at hierarchy and cloud diagrams.

Summary of Building Blocks: The portability of the semantic web infrastructure has been tested against Windows, Linux, and a cloud computing environment. The maintenance of the context model library is abetted by automated features such as web-served browsing of source code and triple-store data elements, site roadmaps and documentation, statistical usage, and password authentication for administration of repository and web server options.

In summary, the integration of a logic processor with triple-store semantic data formatted with RDF and Turtle works effectively as a building block foundation. Using the triple-store for input data files and as a knowledge repository interoperates well with the base logical language. The SWEET ontology provides comprehensive classification and we can build an extensible knowledgebase on top of the SWEET terminology

The domain-specific language for creating stochastic context models includes list processing of n-tuples, math operations on lists, constructors of linear and log range ordinals, dot product and mapping on lists, the latter which we can apply functions and scaling. Specialty functions such as convolution, pair

correlation, Z-difference, integral, derivative, histogram, and simple translation of tuple-lists to graphs make it very convenient for data processing applications.

The domain-specific language for complex-number processing has arithmetical infix operators for multiply, addition, division, power, etc which allows for concise semi-Markov terrain profile modeling, and a built-in FFT for PSD calculation.

The following table lists the categorized building blocks so far described:

Table 1: The set of Domain Specific Language building block elements for environmental modeling covers math and logic functionality

Functionality	Terms	Description	Examples
complex math	isx (evaluation) & (constructor)	Manipulating complex numbers used for generating PSD curves.	`Num isx 1&2 * 2&1`
array math	dot (scalar result) mapdot (array result) ~> (apply function)	High-level array manipulation.	`Z dot Array1 * Array2,` `Amp mapdot sqrt -> PSD * Sx`
statistical	<- (translate to R)	Interface to the R statistics package	`y <- BesselK(1.0,0.0),` `Y <- y`
geospatial	shape point line	Description of geometric shapes and lines	`shape(point(52.3325,4.8673))`
markup	html, table, p, b, body, etc.	Definite clause grammar for generating markup	`table([border(1)], [tr([th('Mean'),th('Sampled')]), tr([td(Mean_Slope), td(S)])])`
semantic	rdf	Library for interfacing to RDF and OWL knowledge	`rdf(U, dcterms:'URI', Link)`
symbolic logic	Prolog terms (including symbols and	Overall logic programming, dimensional checking,	`Distance = 10.0 * meters`

	variables)	code generation translation, etc.	
ordinary math	is (evaluation) Prolog terms	Set of math operators and libraries built-in to the language, used for models and Monte Carlo simulations	`Y is A + B` `random(Sample)`
chart plot	library calls	Integrated JavaScript for artifact display	`context_graphing:plot(…)`
directed graph	library calls	Translation of triple-store links to directed graph	`graphviz:context_graph`

One row from this table was not used but has implications for environmental context modeling. The geospatial functionality is very useful for inferring information on locality. For example, say that temperature statistics are not known for a particular area, but data from nearby locations is available. A geospatial engine can aggregate and select data from weather stations in close proximity and use that to interpolate the statistics. This is built into the semantic web library, yet we deferred applying it until a future need arises.

Pattern Architecture

Based on the modeling described in our foundational research [2][3][4], where we discovered and collected patterns in various environmental contexts, we can start to encode these patterns to build up a comprehensive suite of interactive context models.

We first recognize that many of the context models have similar representation or archetypes, so for example, we can assert that PDF patterns use functional mapping on lists. And we also note that similar representational formats allow reusable artifacts in the form of plots. The basic application of these patterns results in an organization hierarchy shown in **Figure 9**, where we can apply the same development abstractions to each of the contexts that we wish to include. In this case, we consider a terrain context, and realize the same patterns of usage for modeling, language, reasoning, and metrics apply. This may seem overly pedantic, but the patterning approach applies to the development[16] just as it does to the model characterization[17], and it allows the framework to grow and scale. Declarative programming further facilitates pattern-matching, as a set of declarative rules can pattern match automatically on new information.

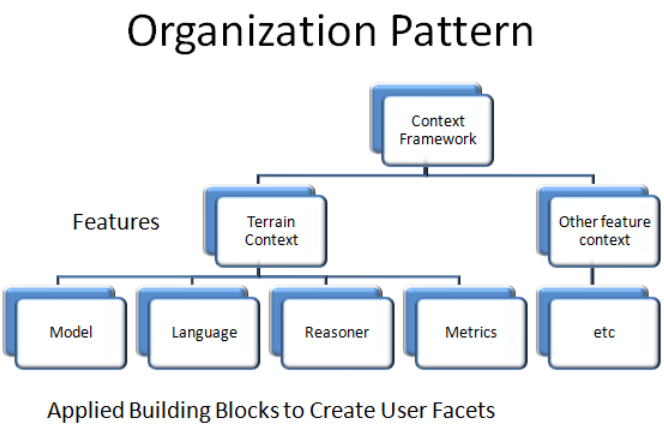


Figure 9: *Pattern of associating domain features with building blocks which allow the creation of user facets and web services, maximizing reuse and maintainability.*

An example of a basic semantic pattern that we can apply is to parameterize models for different geospatial locations (or locales, for short). One locale may be labeled as "**conus**" (shorthand for the continental USA), and used as a semantic node to attach various properties and characteristics to. **Figure 10** below shows the semantic closure graph for how the information gets attached to a locale node. Note that each graph edge represents a predicate in terms of a (*subject*, *predicate*, *object*) triple-store, so that the labeled arrow is the *predicate*, the source of the arrow is the *subject*, and the destination of the arrow is the *object*. This then creates a directed graph of triple-store links.

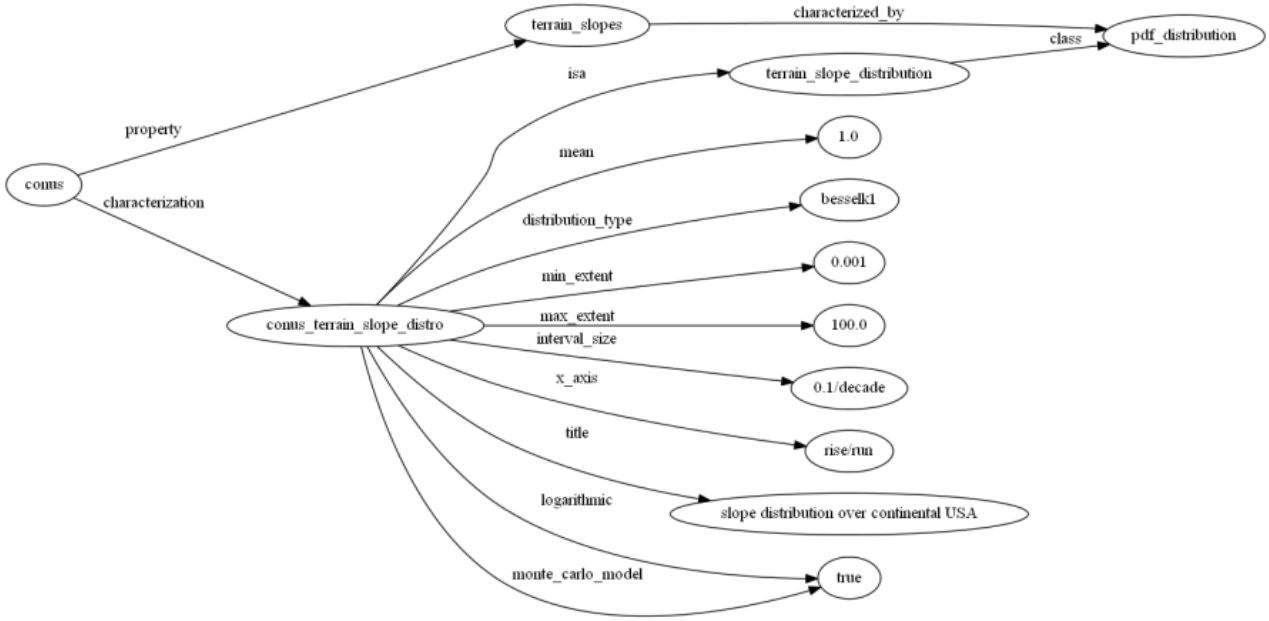


Figure 10 : *Semantic closure graph of triple-store relations associated with a locale "conus" and terrain slope properties. Most rules follow a similar scoping closure.*

This approach allows us to create a pattern architecture based on semantic links.

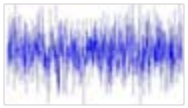
Archetypal Domain Features: The *domain features* of the architecture are the environmental contexts that we are interested in modeling. The following iconified list provides the basic categorization levels.

Fine Terrain Features — Models for representing power spectral densities (PSD) based on Markov and semi-Markov representations. Models of obstacle courses and data sets for friction and soil types.

Gross Terrain Features — Simple distribution for sampling likelihood of terrain slopes. Marginal distributions for elevation changes over various geographic locations.

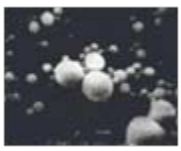
Wave Energy Statistics — Simple approach for modeling PDFs of wave frequencies and wave heights. Models of sea-state wave distributions over various geographic locations. Water density and buoyancy tables.

Wind Energy Statistics — General results for modeling wind energy distribution and autocorrelations for temporal persistence.



EMI Clutter Modeling — Maximum entropy models for electro-magnetic interference (EMI) and noise in the environment. Models of lightning rate.

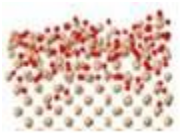
Inland-water Statistics — Maximum entropy estimation for lake size distributions and river flow rates over various regions.



Particle Size Statistics — General method for modeling size distribution of particulates such as ash and ice crystals, and water droplets.

Thermal Dispersion — Simplification of thermal diffusion model to account for disorder and variability in the media. Seasonal and daily temperature models across geospatial regions.

Rainfall Statistics — General model for rainfall distributions within storms using composite process. Cloud size statistics.



Corrosion and Oxidation — Model of oxidation and corrosion growth which improves on the Deal-Grove formulation, using reversion-to-the-mean diffusion kinetics. Table of corrosive categories.

Information Resources: A number of commonly used properties and physical constants span several of these domain features. These can include:

- Physical constants such as water density, gravitational constant, etc.
- Lookup tables such as soil types, sea-state definitions,, etc.
- Nominal environmental and climate information such as standard atmosphere, solar insolation models, etc.
- Dimensional units such as SI and English
- Geospatial definitions and schema – such as UTM and Latitude/Longitude coordinate definitions
- Meta-information: Citations and references for models, algorithms, requirements
- Project specific context data sources
    - Verification & Requirements documents such as TOPS (Test Operating Procedures)[18][19][20]
    - Earth sciences archives such as NASA JPL PO.DAAC [21], US Army Corps of Engineers WIS [22], etc

Archetypal Usage Facets: The usage patterns follow an orthogonal axis that we refer to as usage facets.

Search — The discovery of models needed for a particular purpose is facilitated by various forms of search. A free-form search into the knowledgebase is provided by the search bar in the upper-right corner of the user-interface. This links to knowledge contained within the triple-store knowledgebase, largely independent of semantic context. Other more directed, semantically-driven searches are available from the main search page. Links between specific categories of knowledge and models available within the server are contained here. To accommodate this, specific models are tagged and allocated to specific environmental categories. This is aided by the application of ontologies such as SWEET.

Browse — Environmental and context knowledge follows a natural hierarchical organization. At the top level, we can break out the models into broad categories for Land, Atmosphere, and Aquatic. Below that level, the specific models are allocated to more finely refined categories such as terrain roughness. The basic hierarchy follows that of the SWEET ontology.

Workflows — A process workflow is defined as a software-guided navigation to problem solving. A composable workflow allows for a sequence of problem solving steps depending on the knowledge available. Several workflows to access probability density function (PDF) environmental models and power spectral density (PSD) models are provided. A supplemental system called OSCAR (Ontological System for Context Artifacts and Resources) provides semantic web discovery, registry, and

editing/modification capability. The OSCAR workflow portal will facilitate users to find context models and associated metadata to enable their simulation (see Appendix E).

References — Each model has supporting documentation in the form of references and citations. We also distinguish between specifications, foundational research, requirements, and general reference material. The Zotero citation management system[23] is used to keep track of references and links and we use the SWEET ontology to tag the references with semantic environmental categories. For example, references relating to aquatic wave energy will get tagged with the *phenWave:GravityWave* class resource defined in SWEET.

Resources —Context modeling resources include interactive links to tables and supporting documents, such as environmental regulations, standards, specifications, and nominal or typical operational profiles.

Map View — The scope of environmental models is world-wide. This of course implies that environmental models will depend on the particular geologic and climatic characteristics of specific geospatial locations. Where models have locality links we can search regions of interest to find what is available.

Generic Query — The knowledgebase has SPARQL and native Prolog query support. The intent is that a user can explore the database interactively and reuse resources from the triple-store in other applications.

User Factory — Project-specific requirements provide a means to connect practical applications to information available from an environmental context library. For example, a project requirement that states that a vehicle should be able to operate on terrain of a specific roughness, suggests a link to certain context models available from the server. The links between the requirements and models are accomplished via semantic and ontological organization of the knowledge. In this case, certain keywords and phrases in a requirements document are tagged and allocated to specific environmental categories. This is aided by the application of ontologies such as SWEET.

Dynamic context server knowledgebase and reasoner

The fundamental triple-store data structure generates a substantial fraction of the knowledgebase for the semantic web server. Triple-stores as realized by RDF data sets provide the flexibility to chain and link data at the granularity that we are interested in for context modeling.

The dynamic context server provides interactive content for environmental modeling on demand (i.e. essentially “serving” content). The models are contained within an ontologically organized structure and so can be semantically accessed. The semantic organization allows various search mechanisms, while the information and terminology domains are grouped in data stores according to the bubbles shown in **Figure 11**below.

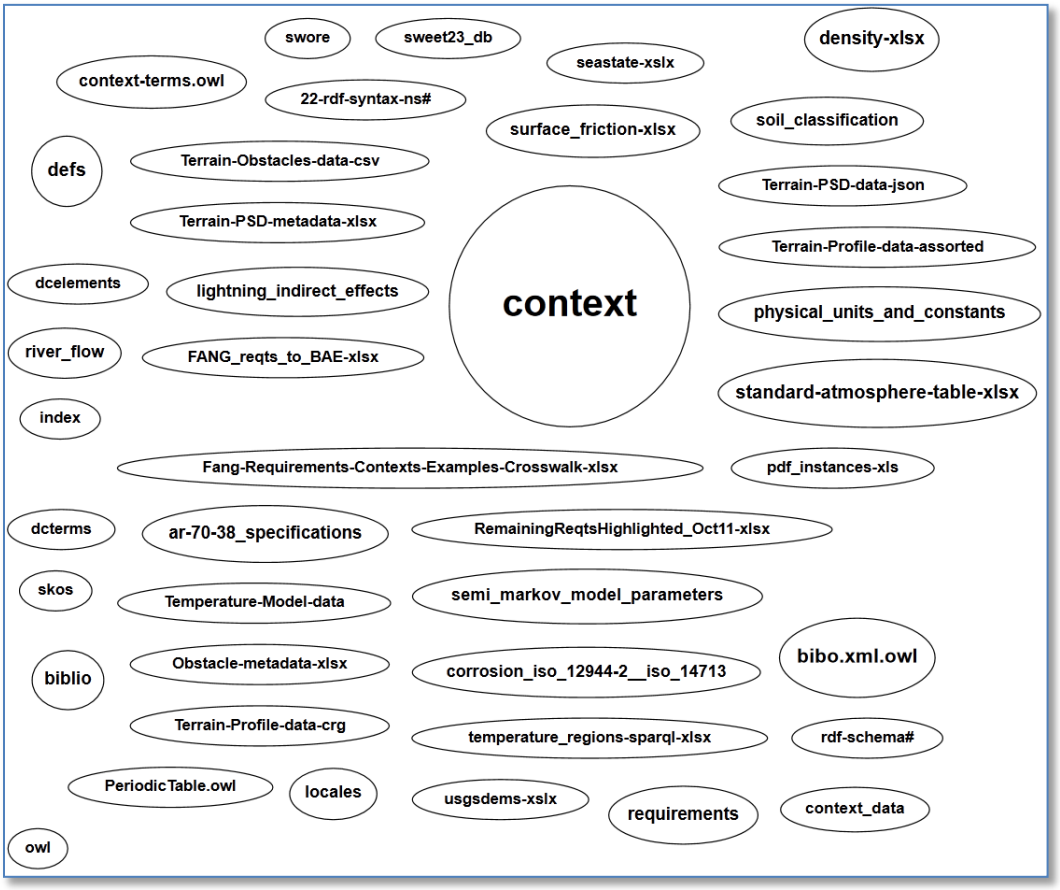


Figure 11: *Cloud describing semantic data sets. The large circle labeled "context" contains the majority of the reference and citation data. Each ellipse corresponds to an RDF or Turtle triple-store data set*

A context is defined as the surrounding environment for a specific intended use, such as may be applied to evaluating a vehicle design or its performance. The environment itself is wide-ranging so that individual contexts can be independently isolated and treated as domain features. This means that a domain feature such as fine-grained terrain can be isolated from the atmosphere domain or, more subtly, distinguished from the overall gross topography of a region. This has important ramifications in that we can encapsulate certain aspects of the environmental models without leaking abstractions across domain boundaries.

Further, as described in **Figure 1**, many of the environmental models are passive and *non-compliant* in terms of interactions with a subject under test. A *compliant* interface would be, for example, a terrain surface that demonstrates a significant give when interacting with a vehicle [24]. These kinds of compliant interfaces demand a more intimately coupled multi-physics interaction between the environment and subject, and is out of the scope of what a semantic context web server can offer in computational throughput and bandwidth. However, the set of passive interfaces remains quite significant and important in evaluating fitness-for-intended-use and other criteria for designs.

Example of Environmental Requirement

As an example of user requirements, consider the paces that a ground-based vehicle has to go through. A set of use-cases may involve the capability to traverse a cross-country course with a specific root-mean-square (RMS) deviation in the terrain roughness. If requirements phrases such as "cross-country" and

"RMS ride courses" are captured in a semantic knowledgebase, rules can then be added to automatically link to a specific environmental model. This is shown in **Figure 12** below.

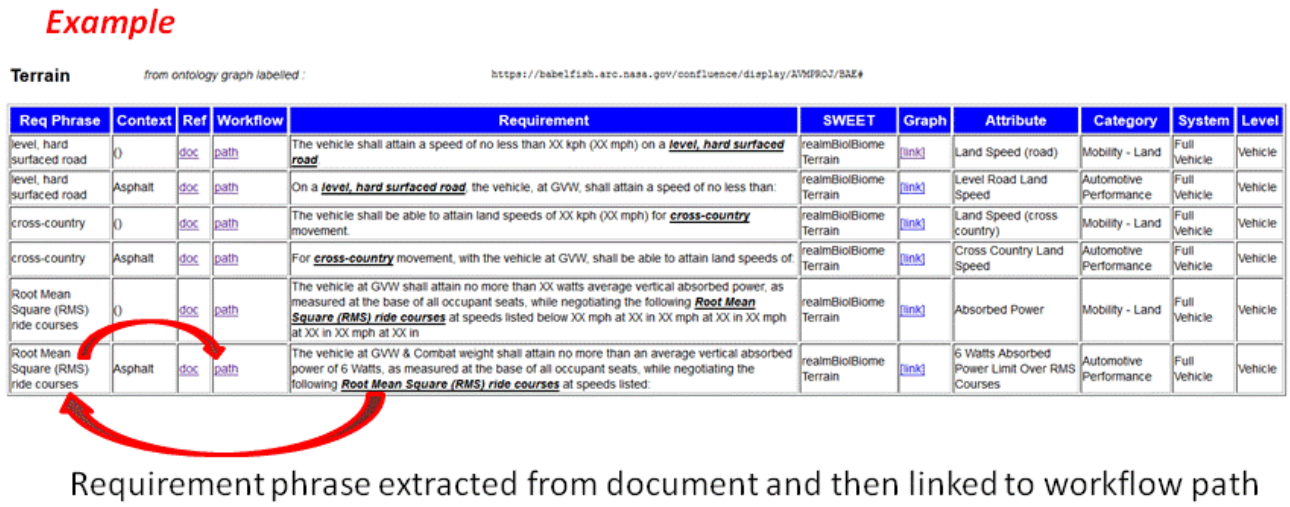
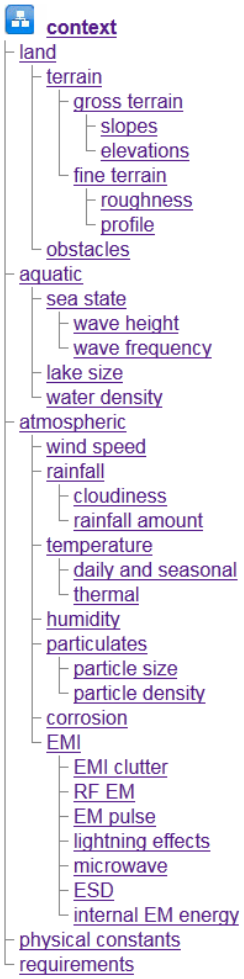


Figure 12: *Requirements to workflow*

If these phrases are captured beforehand and stored as persistent knowledge with links to categories, models and workflows, we can take advantage of "factory-level" automation when a new project requirements document is introduced. Individual requirements stored as triples within a data store (see the ellipses labeled in "Fang-req..." and "RemainingReqts…" in **Figure 11**) can be parsed by the expert system with the phrases highlighted within individual requirements. A tool such as OpenRefine[25] can help with the translation from a spreadsheet to a triple-store format such as RDF[26], N3, or Turtle.

In terms of a semantic web interface, the user is presented with a listing of phrases discovered within a semantic context, and then tabulated against a path to the appropriate workflow.

An aspect to this formulation guards against potential requirements churn. As new requirements come in with extra decomposition levels, new triple-store linkages can be added without upsetting any previous triples that may have previously existed. In other words, a perfect relational schema does not have to be maintained, which is an issue when applying the output from typical requirements databases. Adaptability of knowledge format is the key here, while keeping a balance between free-form textual association and tacit semantic relationships.



Browsing Interface

The server provides a browsing interface that is structured around the natural hierarchy that we employ of *land*, *aquatic*, and *atmospheric* domains.

Internally, we employ triple-store relationships to manage the links in the hierarchy. The hierarchy (see right) is extended to provide space for resources

that may span features, see *physical constants*, and for those that may be user defined, *requirements*.

See the Annex A for a browser-guided narrative that is provided as part of the dynamic context server. This explains the rationale for the categorization chosen for the set of models.

Reference and Citation Linking

The context knowledgebase contains semantic information which allows it to be searched in a structured fashion. The set of references to research articles, specifications, and standards is managed by the Zotero documentation citation system [27][28][23]. Individual citations are tagged with terms from SWEET within the Zotero browser component (as a Firefox plug-in) to indicate the environmental concepts that apply.

For non-structured information retrieval, the search bar in the upper-right corner finds information within the knowledgebase that is not semantically organized.

A typical semantically-based query is shown in **Figure 13**, shown below for a specific SWEET term signifying the *Temperature* property class. Regular references are distinguished from specifications ( ) and model foundational documents ( ), the latter of which contain detailed information to support the construction of models for the context library.

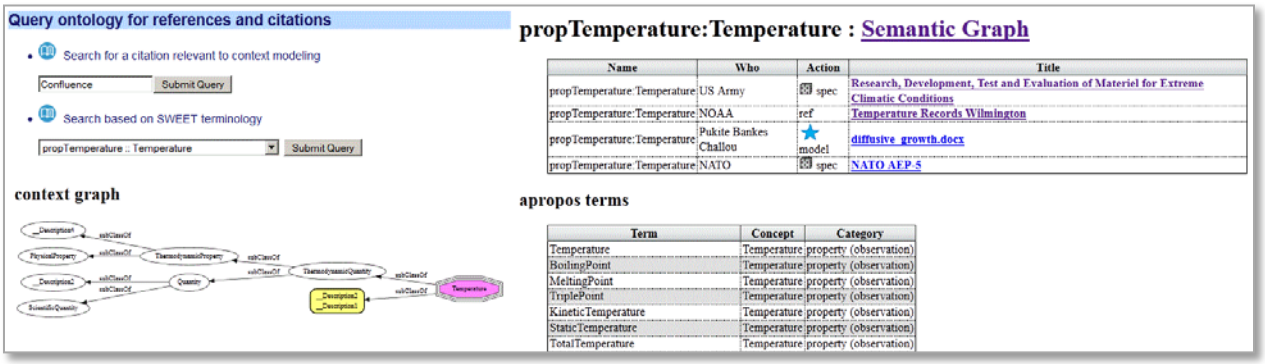


Figure 13 : *View of the reference and citation search interface.*

Capturing and integrating reference and citation knowledge via a standard citation management system and a standard ontology such as SWEET allows for future growth, aided by the fact that other users can contribute knowledge to the reference database.

Workflows

Workflow patterns follow the concept of economized development. We rely on patterns of workflow to reduce modeling work, simplify maintenance, and create a record of a project's provenance.

The supplemental workflow used to access statically defined context models for test-bench execution is called OSCAR (*Ontological System for Context Artifacts and Resources*).

OSCAR is a knowledge-based system providing semantic web discovery capability, limited according to the available context library ontology and instance data (see Appendix E). The goal is to provide guided discovery for users to find, register, or edit context models and associated metadata to be used in their simulation. The context models include collections of power spectral density profiles (PSDs) and other

data sets, which provide the generators for model execution. This is considered an industrial strength component as it has a design lineage tracing to oceanography data content service architectures [29].

The other workflows available are intended to provide dynamic content for probabilistic evaluation, such as what is needed to perform PCC (probabilistic certificate of correctness)[1] and for support of formal verification. In these cases, probability density functions provide the constraints and likelihoods for testing extreme values or corner cases, with the distributions mathematically provided to allow Monte Carlo simulations to explore the required state space for test cases.

The workflows can also be used to quickly evaluate model characteristics and collect artifacts for later decision making. For example, the earlier example of requirements search is actually a guided workflow that starts from the captured requirements phrases.

A sampling of the workflows available is shown in Figure 14 below:

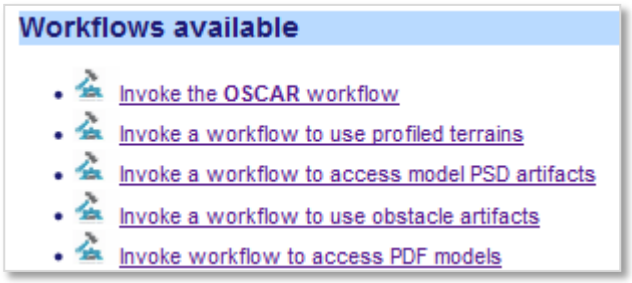


Figure 14 : Workflows available

Search

Many common environmental models share the same underlying stochastic pattern. In general, this means that we can use similar math to capture the characteristics of behavior across a range of domains, even though the measured quantities are not comparable.

The important case for PCC estimates is the probability density function (PDF) of environmental characteristics showing a spread in uncertainty in values.

PDFs and other models can be searched according to feature keywords or SWEET categories based on applicability to the context space as shown in **Figure 15**.

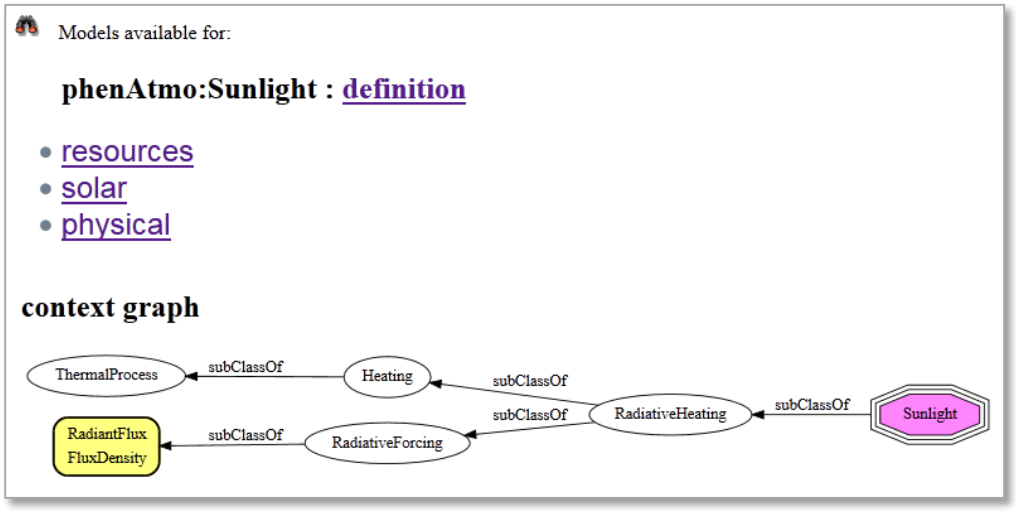


Figure 15 : *Search results for a SWEET classification term*

The general rule we apply for general content search is to base it on ontological terms (if not a free-form textual phrase) and then use that to restrict the resources that match.

The presentation of knowledge from the matching hits could be in HTML with links such as shown in **Figure 15** or in XML, JSON, CSV or other format suitable for external use.

The server also provides a more free-form textual search that evaluates contents of the triple-store, such as `rdfs:comment` instances, and then presents the results as a list of triples (see **Figure 16**):

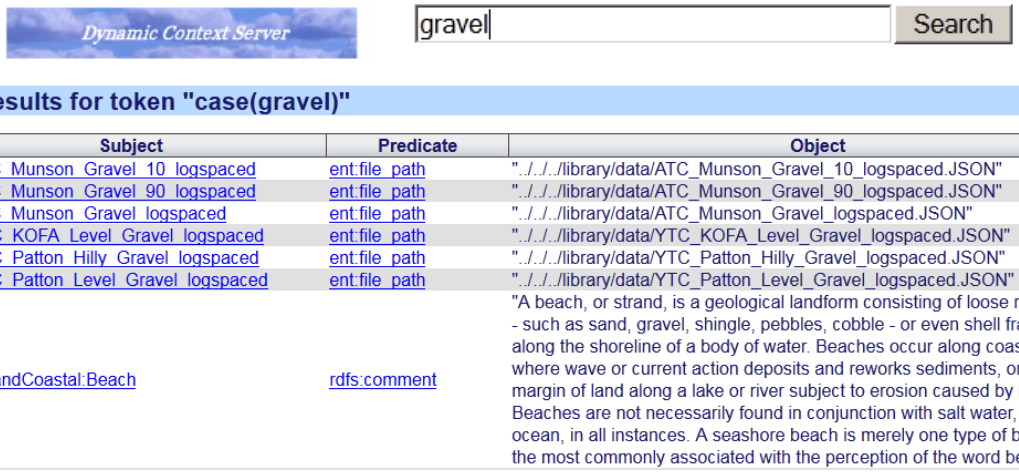


Figure 16*: Free-form search partial results for the phrase "gravel".*

Resources

Models in the form of interactive standards are also provided via the context server. These are transcribed into semantic content from various standards documents. The semantic content enables the data to be used to automatically construct tables, graphs, or other structural forms (see **Figure 17**).

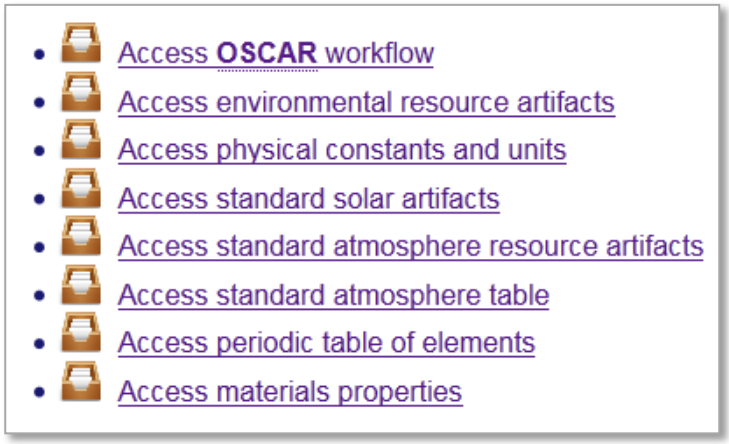


Figure 17 : *List of resources (tables, charts, and data) taken from environmental standards.*

Also units of dimensionality and physical constants are categorized here. For various dynamic artifacts, the applicable units are stored as resources and necessary conversions between units are done automatically.

Examples of tables from triple-store include the Periodic Table of Elements (**Figure 20**), a Coefficient of Friction table (**Figure 19**), and a Soil classification table (**Figure 18**).

category	division	groupSymbol	groupName
coarse grained	gravel	GW	gravel containing well-graded (diversified particle sizes)
coarse grained	gravel	GP	gravel containing poorly graded (uniform particle sizes)
coarse grained	gravel	GC	gravel containing clay-like character
coarse grained	gravel	GM	gravel containing silt-like character
coarse grained	sand	SW	sand containing well-graded (diversified particle sizes)
coarse grained	sand	SP	sand containing poorly graded (uniform particle sizes)
coarse grained	sand	SC	sand containing clay-like character
coarse grained	sand	SM	sand containing silt-like character
highly organic	organic	OH	organic high plasticity mainly clay-like character
highly organic	organic	OL	organic low plasticity mainly silt-like character
fine grained	clay	CL	clay with low plasticity
fine grained	silt	ML	silt with low plasticity
fine grained	clay	CH	clay with high plasticity
fine grained	silt	MH	silt with high plasticity

Figure 18: *Soil Classification table featuring USCS two-character code automatically*

The following coefficient of friction table **Figure 19** was generated from a table of values captured from engineering data[30] and then converted to RDF triples through the *OpenRefine* tool[25].

surface material	description	dry slow friction	dry fast friction	wet slow friction	wet fast friction	constraint	note
PORTLAND CEMENT	New,Sharp	[.80, 1.20]	[.70, 1.00]	[.50, .80]	[.40, .75]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
PORTLAND CEMENT	Traveled	[.60, .80]	[.60, .75]	[.45, .70]	[.45, .65]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
PORTLAND CEMENT	Traffic Polished	[.55, .75]	[.50, .65]	[.45, .65]	[.45, .60]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
ASPHALT or TAR	New,Sharp	[.80, 1.20]	[.65, 1.00]	[.50, .80]	[.45, .75]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
ASPHALT or TAR	Traveled	[.60, .80]	[.55, .70]	[.45, .70]	[.40, .65]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
ASPHALT or TAR	Traffic Polished	[.55, .75]	[.45, .65]	[.45, .65]	[.40, .60]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
ASPHALT or TAR	Excess Tar	[.50, .60]	[.35, .60]	[.30, .60]	[.25, .55]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
GRAVEL	Packed,Oiled	[.55, .85]	[.50, .80]	[.40, .10]	[.40, .60]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
GRAVEL	Loose	[.40, .70]	[.40, .70]	[.45, .75]	[.45, .75]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
CINDERS	Packed	[.50, .70]	[.50, .70]	[.65, .75]	[.65, .75]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
ROCK	Crushed	[.55, .75]	[.55, .75]	[.55, .75]	[.55, .75]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
ICE	Smooth	[.10, .25]	[.07, .20]	[.05, .10]	[.05, .10]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
SNOW	Packed	[.30, .55]	[.35, .55]	[.30, .60]	[.30, .60]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast
SNOW	Loose	[.10, .25]	[.10, .20]	[.30, .60]	[.30, .60]	Not for heavy large trucks	Speeds of 30 MPH separate slow from fast

***Figure 19**: Coefficient of Friction table automatically constructed from semantic elements*

Perhaps the most sophisticated semantic-based generation example is the periodic table of elements shown below, which uses supplemental logic rules for element placement and coloring.

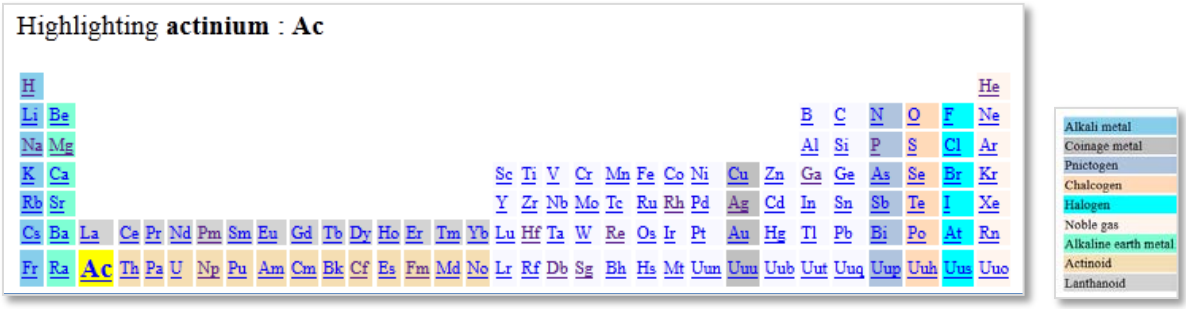


***Figure 20 :** Periodic Table of the Elements generated from semantic rules*

Examples of generating graphs from triple-store include rendering of ideal gas curve and nominal temperature specifications for various climactic conditions (see **Figure 21**). These are taken from the environmental modeling handbook AR-38 [31].

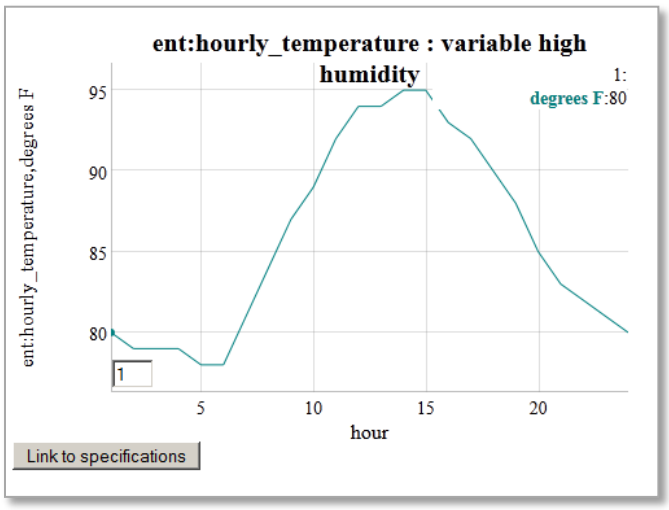


***Figure 21 :** Generating a plot from an environmental specification of nominal daily humidity swings.*

Map / Location

Environmental models can either be general in terms of their suitability (such as water properties or a standard atmosphere), or specific to a geospatial location, which can further include daily or seasonal effects.

The locations are accessible through the semantic drop-down list or through indexed search terms with keystroke look-ahead.

Figure 22: Keystroke look-ahead for domain features linked to geo-spatial locations.

The map button indicates the location of the model or data, and the available button links to the modeling domain(s) available. A Google Map API window is also provided to illustrate the geospatial location and coordinates of the model or data. Both these features are shown in **Figure 23**.

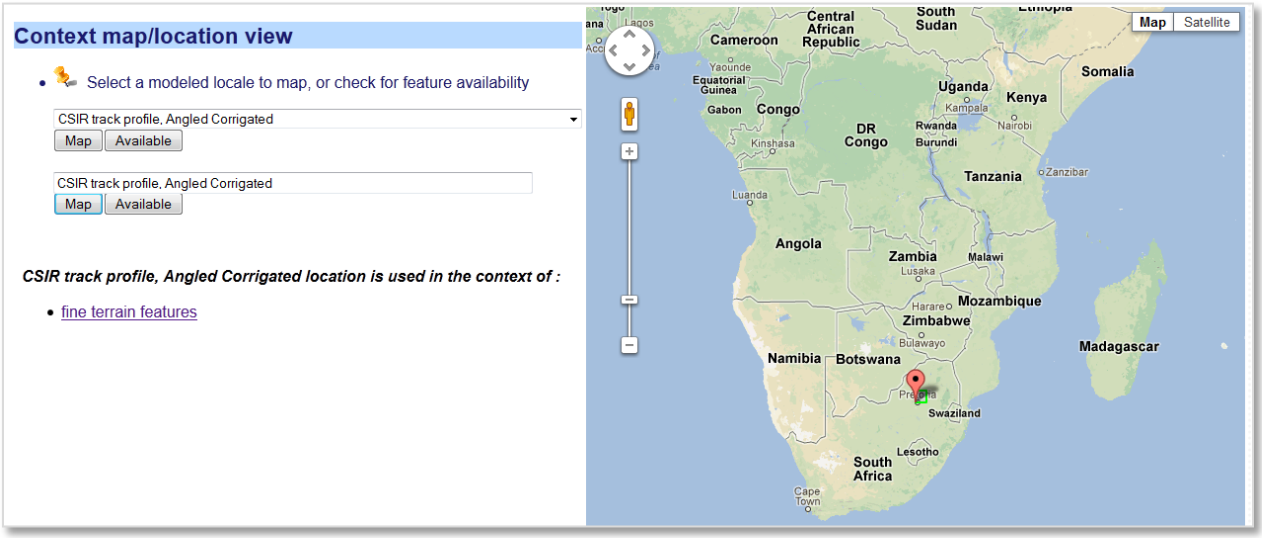


Figure 23: *Google Map window for locating geospatial location of queried model, with link to model on left.*

Knowledgebase and Server Technical Architecture

The internal structure of the dynamic contest server consists of a semantic reasoning engine and its accompanying set of knowledge. The interactivity with this underlying architecture via a web server allows for a user interface and potentially automated modes (see **Figure 24**) to remote data sources or to test benches.

The ontology is cast as a form of *description logic* [1], which consists of a terminology set (the so-called terminology box or *TBox*) and an asserted data store (the assertion box or *ABox*). The terminology and assertions can come from standard ontologies such as SWEET (for the TBox) and from qualified data and information sources (for the ABox). In other words, the former provide latter provides the classification and the latter provides knowledge and data.

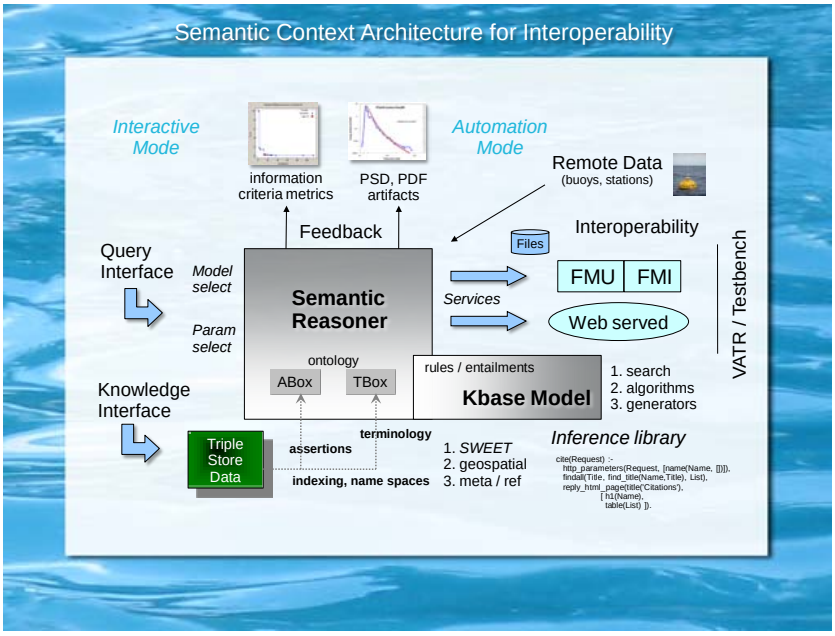
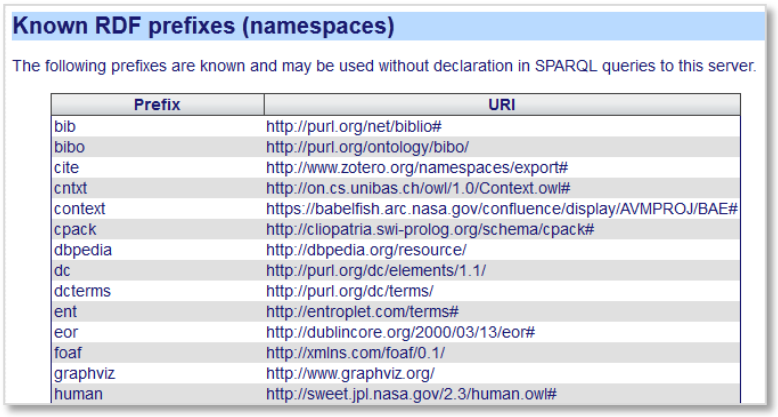


Figure 24: *Usage architecture for the semantic web server. The Dynamic Context Server handles artifacts and interactive services, while OSCAR handles static data and executables such as functional mockup units (FMU)*

The logical rules work with the terminology and assertions to accomplish searches, execute algorithms, generate code, and otherwise perform general inferencing.

To interface concisely with the triple-store database, a set of *namespace prefixes* is specified and used by the reasoner. A subset of the prefix namespace is shown below in **Figure 25**. As the SWEET ontology provides a comprehensive view of natural terminology, this by necessity has the largest set of prefixes, largely to provide disambiguation for terminology that is borrowed across domains. For example, a *wave* could refer to an aquatic wave or an electromagnetic wave, with the namespace helping to provide a context.

The namespace called *ent:* holds special significance as it provides sub-classing and additional terminology to the SWEET ontology. By providing an additional level of abstraction, we can guard against changes made to the reused standard ontologies.



Known RDF prefixes (namespaces)

The following prefixes are known and may be used without declaration in SPARQL queries to this server.

Prefix	URI
bib	http://purl.org/net/biblio#
bibo	http://purl.org/ontology/bibo/
cite	http://www.zotero.org/namespaces/export#
cntxt	http://on.cs.unibas.ch/owl/1.0/Context.owl#
context	https://babelfish.arc.nasa.gov/confluence/display/AVMPROJ/BAE#
cpack	http://cliopatria.swi-prolog.org/schema/cpack#
dbpedia	http://dbpedia.org/resource/
dc	http://purl.org/dc/elements/1.1/
dcterms	http://purl.org/dc/terms/
ent	http://entroplet.com/terms#
eor	http://dublincore.org/2000/03/13/eor#
foaf	http://xmlns.com/foaf/0.1/
graphviz	http://www.graphviz.org/
human	http://sweet.jpl.nasa.gov/2.3/human.owl#

Figure 25 : *Configured prefixes for URI name-spaces.*

As an example of the sub-classing, the resource **ent:C4** refers to a specific classification of corrosive environment (see **Figure 26**). According to the semantic hierarchy, it sub-classes from both the SWEET **procChemical:Corrosion** class as well as the **procCategorical:StandardIndustrialClassification** class from SWEET. The directed graph also shows how these SWEET classes are sub-classed within that hierarchy. The C4 property is extended to include specific information relevant to this corrosion classification, such as qualitative descriptions of **ent:environmental_conditions** and **ent:exterior_examples**.

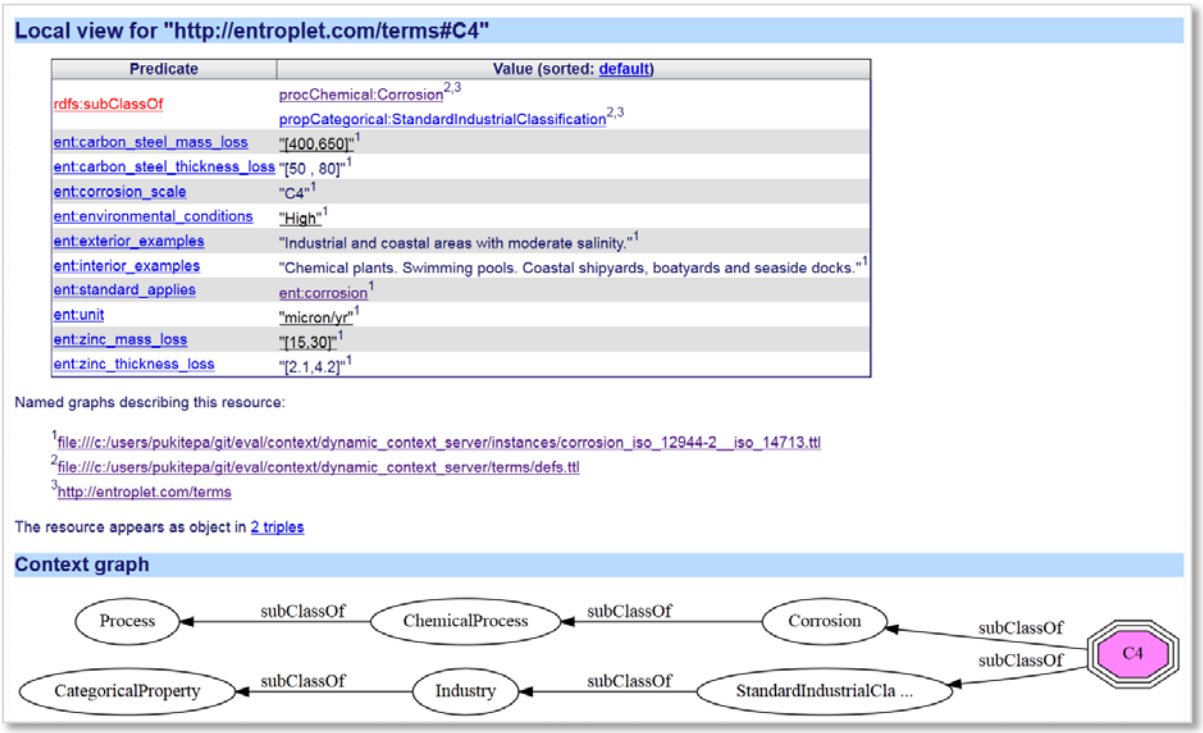


Figure 26: *Local view of the ent:C4 resource represented as an octagon-shaped node*

To understand how this resource can be used, refer to the corrosion classification table in **Figure 27** shown below. This is constructed based information contained in ISO standards [32][33] and then converted to triple-store. The semantic rules then enable the generation of the table.

class	env	exterior	interior	carbon steel	zinc	units
C1	Very low	Inland rural	Interior of buildings, offices, stores, schools, hotels. Indoor spaces with occasional condensation.	[1,3]	[0,0.1]	micron/yr
C2	Low	Very ligh industrial zone or in a city without pollution	Warehouses, sports auditoriums, logistic centers, hangars. Dry indoor spaces.	[3,25]	[0.1,0.7]	micron/yr
C3	Medium	Inland urban, mildly saline. Urban and Industrial atmospheres, moderate SO2 pollution, coastal areas with low salinity.	Indoor spaces with high moisture content, not much impurities. Production rooms with high humidity and some air pollution, e.g. food processing plants, breweries, laundries.	[25 ,50]	[0.7,2.1]	micron/yr
C4	High	Industrial and coastal areas with moderate salinity.	Chemical plants. Swimming pools. Coastal shipyards, boatyards and seaside docks.	[50 , 80]	[2.1,4.2]	micron/yr
C5-I	Very high (heavy industrial zone)	Very humid industrial atmosphere. Heavy industrial zone, contact with chemicals .	Industrial plants, warehouses, heavy equipment workshops	[80,200]	[4.2,8.4]	micron/yr
C5-M	Very high (marine atmosphere)	Saline seaside atmosphere. Coastal and off-shore areas with high salinity, up to 1000 m from the coastline	Buildings or areas with almost permanent condensation and high pollution.	[80,200]	[4.2,8.4]	micron/yr

Figure 27: *Corrosive classification table based on ISO 12944-2 and ISO 14713 constructed from semantic elements*

The reasoner itself uses rules and logic that are embedded in run-time loaded modules. A partial list of logic modules is shown in **Figure 28**.

Server plugin configuration

Config	Title	Status
010-packs		Local
020-prefixes	Configure prefixes (namespaces)	Installed (modified)
100-used_modules		Local
cache	Configure caching of RDF inputs	Not installed
cloud		Installed (linked)
con_text	HTML utilities	Local
config		Installed (linked)
context	Top-level context module	Local
context_acronyms		Local
context_ar7038	Environmental specifications according to standard	Local
context_atm	Atmosphere specifications	Local
context_autocorr	Topography interface	Local
context_browse	Tree browser for context models	Local
context_clutter	Model of EMI clutter	Local
context_codegen	Code Generation utilities	Local
context_complex	Math operations for complex numbers	Local
context_corrosion	Models of corrosion	Local

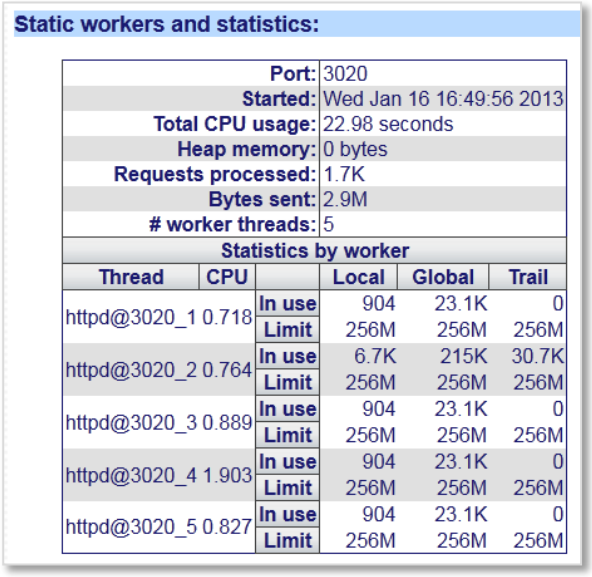
Figure 28: *The web server Prolog run-time automatically loads all available modules specified by a configuration. These are considered the set of server plug-ins.*

The cloud of **Figure 11** uses a manifest of triple-store data-sets. The manifest itself is an RDF list in Turtle format which describes the name and location of each of the triple sets loaded by the server. So the general strategy is to load the logic and data at run-time startup[2].

[2] The data-sets are cached and optimized for faster loading on subsequent start-ups.

The consistency of the knowledgebase is automatically tested on load and reload by a unit-test suite, which is equipped to execute whenever module-based changes are made to the knowledgebase,.

The final deployment is targeted for cloud computing, and the server can be accessed at initially an Amazon EC2 server located at http://entroplet.com. Statistics of usage are accessed from an administration level, with typical output shown in **Figure 29**.



Static workers and statistics:

Port:	3020
Started:	Wed Jan 16 16:49:56 2013
Total CPU usage:	22.98 seconds
Heap memory:	0 bytes
Requests processed:	1.7K
Bytes sent:	2.9M
# worker threads:	5

Statistics by worker

Thread	CPU		Local	Global	Trail
httpd@3020_1	0.718	In use	904	23.1K	0
		Limit	256M	256M	256M
httpd@3020_2	0.764	In use	6.7K	215K	30.7K
		Limit	256M	256M	256M
httpd@3020_3	0.889	In use	904	23.1K	0
		Limit	256M	256M	256M
httpd@3020_4	1.903	In use	904	23.1K	0
		Limit	256M	256M	256M
httpd@3020_5	0.827	In use	904	23.1K	0
		Limit	256M	256M	256M

Figure 29: *Administrative statistical view of server usage.*

Examples of usage

The following are examples of usage for the dynamic context server. These are selected to show a cross-section of capabilities available.

Sea State View

A sea-state model is used to provide a standard quantification for the significant wave height expected or measured for various coastal and inland bodies of water. A sea-state rating follows a scaling system starting at 0 for calm water. The objective and subjective description for a specific sea-state depends on the characteristics of the environment as defined by the World Meteorological Organization specification[34] (see **Figure 30**):

value	low	high	units	description
0	0	0	m	Calm (glassy)
1	0	0.1	m	Calm (rippled)
2	0.1	0.5	m	Smooth (wavelets)
3	0.5	1.25	m	Slight
4	1.25	2.5	m	Moderate
5	2.5	4	m	Rough
6	4	6	m	Very rough
7	6	9	m	High
8	9	14	m	Very high
9	14	9999	m	Phenomenal

Figure 30: *Semantically generated chart of sea-stat rating system*

The wave elevation cumulative distribution function (CDF) maps to the sea-state for a given region, such that lower sea-state values are more common for smaller and shallower bodies of water. The mapping from sea-states to probabilities over a long term is shown in **Figure 31**.

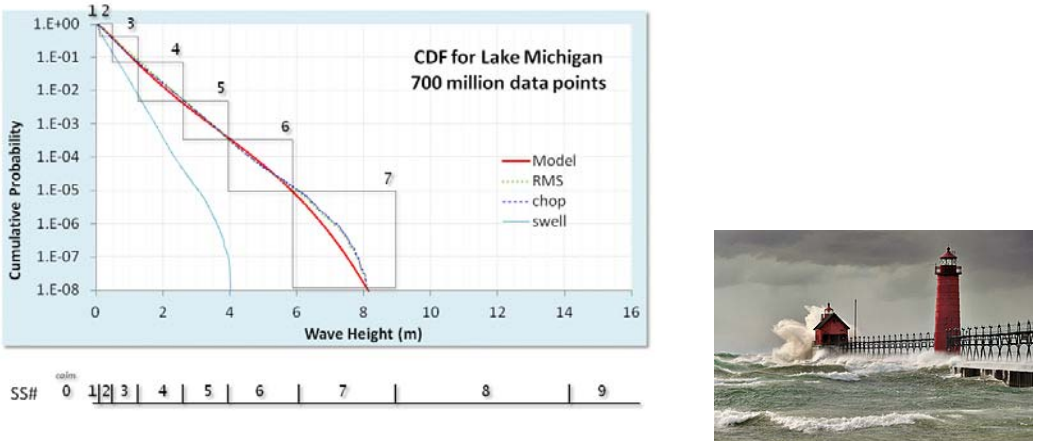


Figure 31*: Sea-state rating mapped to a CDF culled from several years of Lake Michigan significant wave height data. Sea-state classification depends on probability distribution of wave-height according to a specific geo-spatial location. Sea-states of higher than index 7 have not been observed for this body of water.*

To make the sea-state table practical for probability estimates in an interactive setting, we needed to attach the sea-state alignment to a particular region (se **Figure 32**).

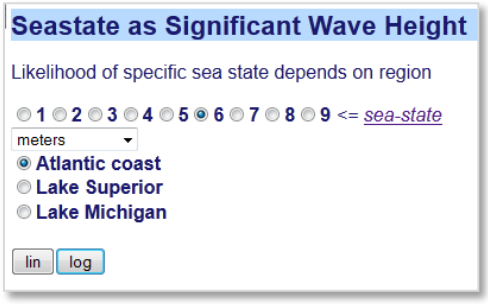


Figure 32: *Sea-state selection window*

If we select a sea-state and the Atlantic coast region with logarithmic scaling, the following cumulative distribution function (CDF) chart results (see **Figure 33**). The reasoner automatically intersects the low and high end of the significant wave height for the sea-state enumeration selected.

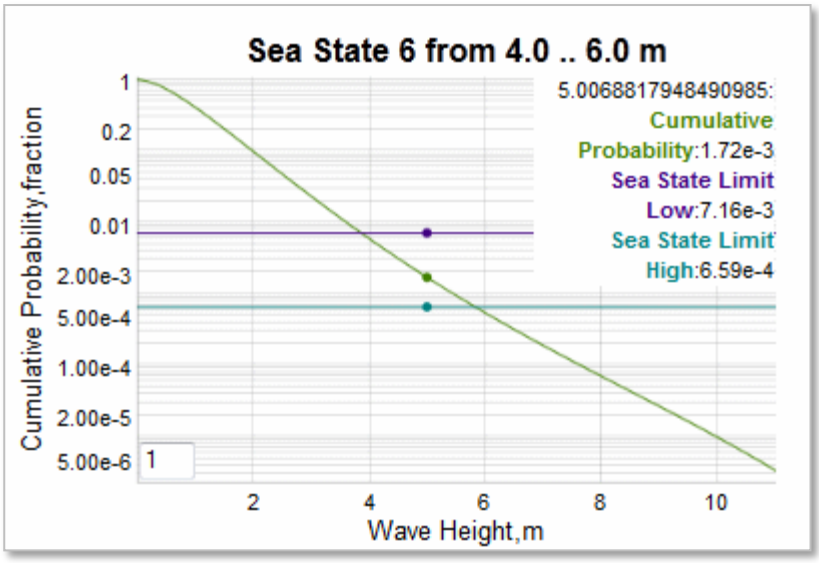


Figure 33: *Selected interactive sea-state 6 classification for Atlantic coast*

In the example chart above, the cumulative probabilities for a sea-state of index 6 are highlighted at the position of the cursor. The lower cumulative is given by the purple line and the upper (rarer) cumulative given by the high-end sea-state value is the bluish-green line.

The ranges depend on a model fit to the historical data culled from the selected aquatic region[2], so to add another region requires only an additional region and parameterized model to the triple-store database.

Obstacles

Not all terrain features have natural origins. We catalogued man-made terrain profiles as *obstacles* and captured and stored their geometric shape as deterministic segments. Most of the obstacle profiles were culled from specifications derived from vehicle test courses[20]. Since these test courses were created specifically for testing vehicles with a parallel wheel-base, we stored the data as double track profiles with a fixed separation between the two tracks.

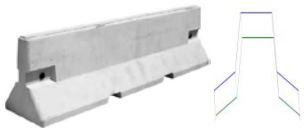


Figure 34: *Jersey Barrier and two-track deterministic profile shown beside it.*

The range of obstacles includes jersey barriers (**Figure 34**), washboard surfaces (**Figure 35**), cross-tie courses (**Figure 36**), half-round bumps, staircases, potholes, and ditches. To get a rough indication of the spatial orientation of a track, a feature for rendering the double tracks as a 3D image is provided. This extrapolates the detailed profile specification as an extruded three-dimensional projection (see **Figure 35**).

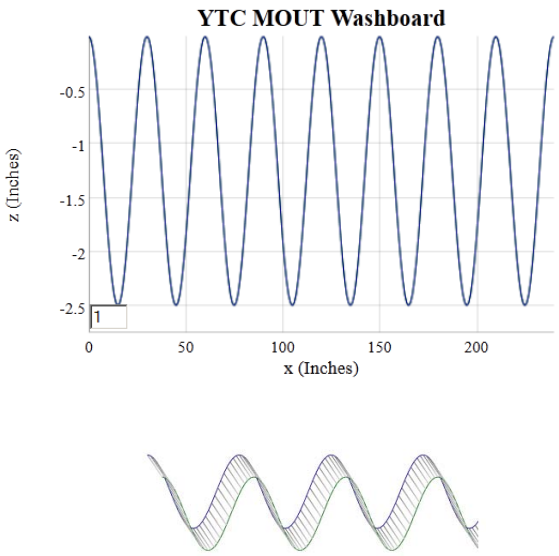


***Figure 35** : Washboard track. Detailed view (above) and 3D view (below).*

A workflow was incorporated to guide the choice of obstacle profile data sets (**Figure 36**). In this case a cross-tie course is rendered in a schematic projection, along with the extrapolated 3D view. The green trace is the inside track and blue trace is the outside/far track of what could represent remnants of an old railroad track.

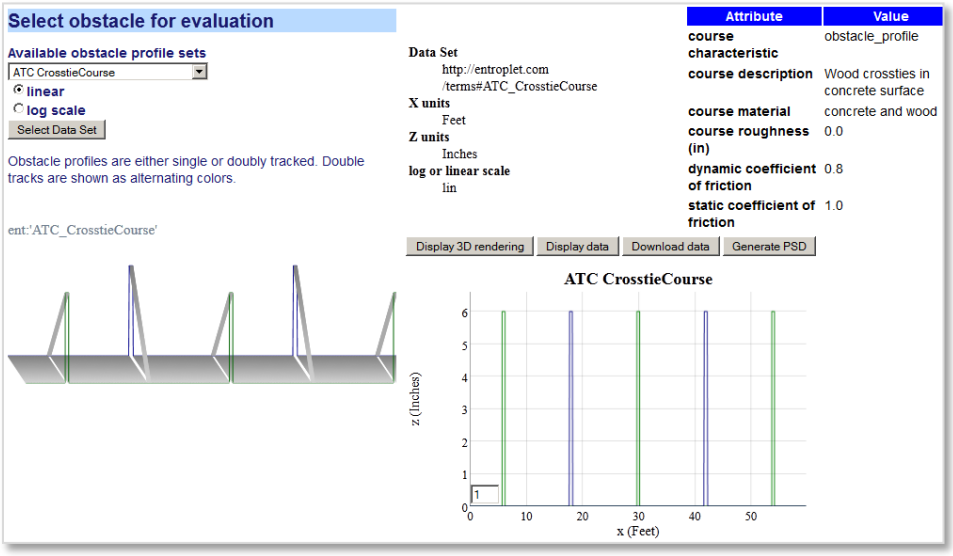


***Figure 36**: Obstacle evaluation workflow showing selection of a cross-tie track.*

The deterministic nature of these tracks is evident when one considers that artificial washboards (or periodic cross-ties) as shown above have a defined periodicity while natural washboards created by vehicular traffic (or decomposing and shifting cross-ties) show a disordered range of periodicities. The latter are more suited to descriptions by a semi-Markov model, which is described next.

Fine-relief Terrain View

We illustrate two complementary approaches to generating terrain profiles based on power spectral densities (PSD).

1. Superposition of randomly phased sine waves to accomplish a pseudo inverse Fourier transform of the PSD. This is essentially a Fourier series reconstruction
2. Fitting a semi-Markov autocorrelation model of random walk to the PSD (via the Weiner-Khinchin theorem [35])

Each approach has benefits and drawbacks

Benefits	▪ The superposition approach is fast and automatic as it works as a rough heuristic ▪ The semi-Markov autocorrelation function approach is based on stochastic properties of the terrain relief changes so can model properties such as phase and skew**[3]**
Drawbacks	▪ The superposition approach will show a repeat sequence based on the lowest spatial frequency and needs to approximate phase (no asymmetries possible) ▪ The semi-Markov autocorrelation function approach requires a fitting process, and gets the most benefit from prior knowledge in addition to that provided by the PSD. However, once the models are created, new models can easily be composed from the old ones.

The terrain profile context models include both variants.

Markov and semi-Markov Processes

The fine-relief profile of natural terrain features can be characterized and modeled effectively by Markov and semi-Markov processes.

Mathematical Markov formulations such as the Ornstein-Uhlenbeck process[36] can model rough random terrain, while semi-Markov processes capture pseudo-random features with emerging periodicities[3]. We provide a workflow to guide the user in evaluating these kinds of models as shown in **Figure 37**.

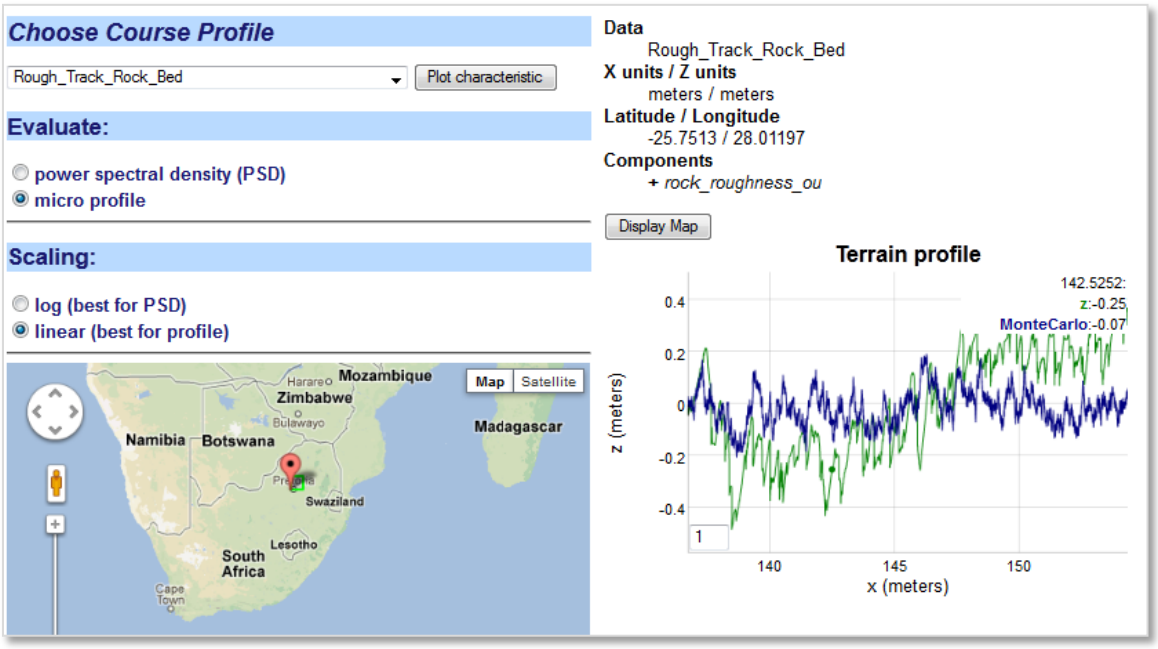


Figure 37: *Micro profile simulation of a rough track (labeled z) overlaid with a Markovian Ornstein-Uhlenbeck random walk Monte Carlo simulation*

To generate a profile, a Monte Carlo simulation draws a sequence of steps from sampling from the Markov or semi-Markov algorithm. This is compared to a profile from a high-resolution terrain taken from the Gerotek vehicle testing course, in this case a rough track rock bed.

For the PSD, the plot consists of two statistically equivalent representations starting from the Markov formulation, and the FFT of the data it is derived from (see **Figure 38**):

- An analytical expression for the PSD derived stochastic representation
- A windowed FFT of the Monte Carlo profile drawn from the stochastic representation.
- The FFT spectrum of the data.

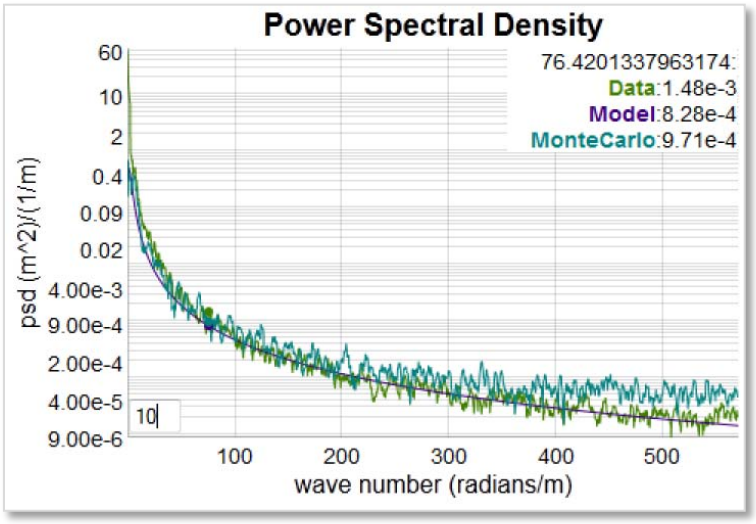


Figure 38*: PSD of rough road*

High-resolution data from CSIR is stored in the database (see the sets labeled GEROTEK and Rough Track) as well as a few other test sets taken from online archives[37].

The FFT can show substantial noise, so we use a moving average filter to reduce the noisy values so that the underlying spectrum is clearly delineated. The graph can be stretched and then filtered to reveal the signal (see **Figure 39**).

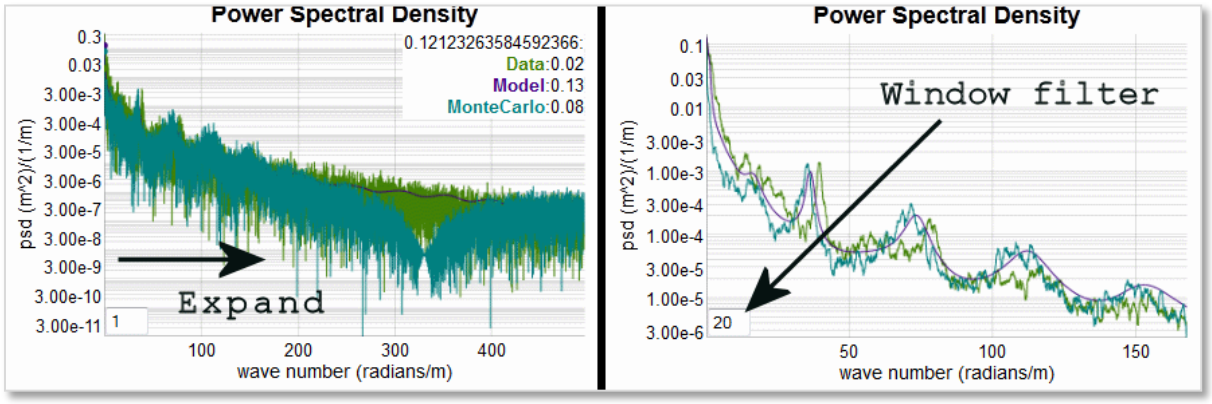


Figure 39: *Use of interactive filter and drag selection to narrow in on a portion of the frequency spectrum. When the filter is used the underlying fine structure used for fitting is revealed.*

The model contains parameters that are stored as a few triples, while the original data is also a triple but one that can contain several thousand data points. The general fitting strategy is shown in **Figure 40** and described in detail elsewhere[3].

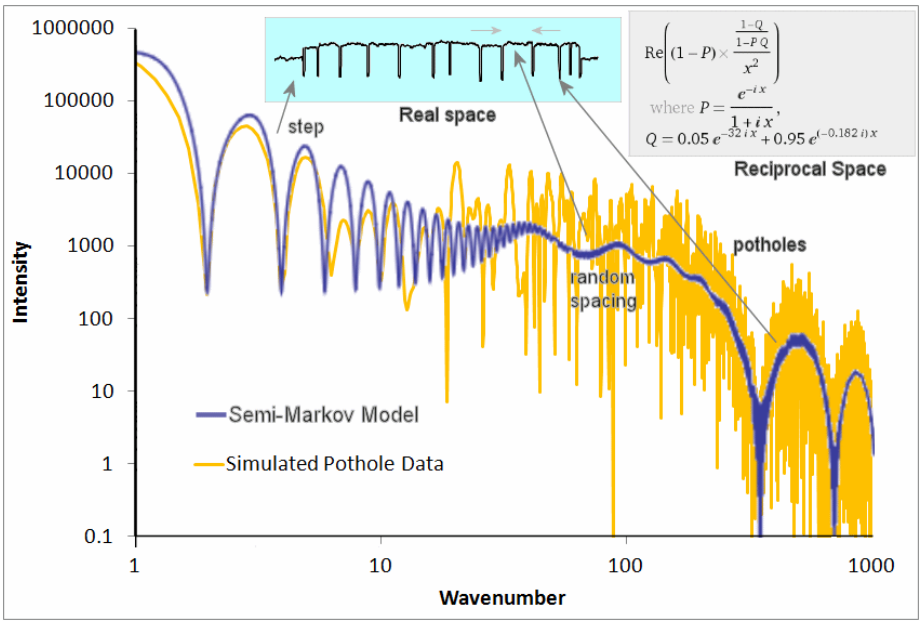
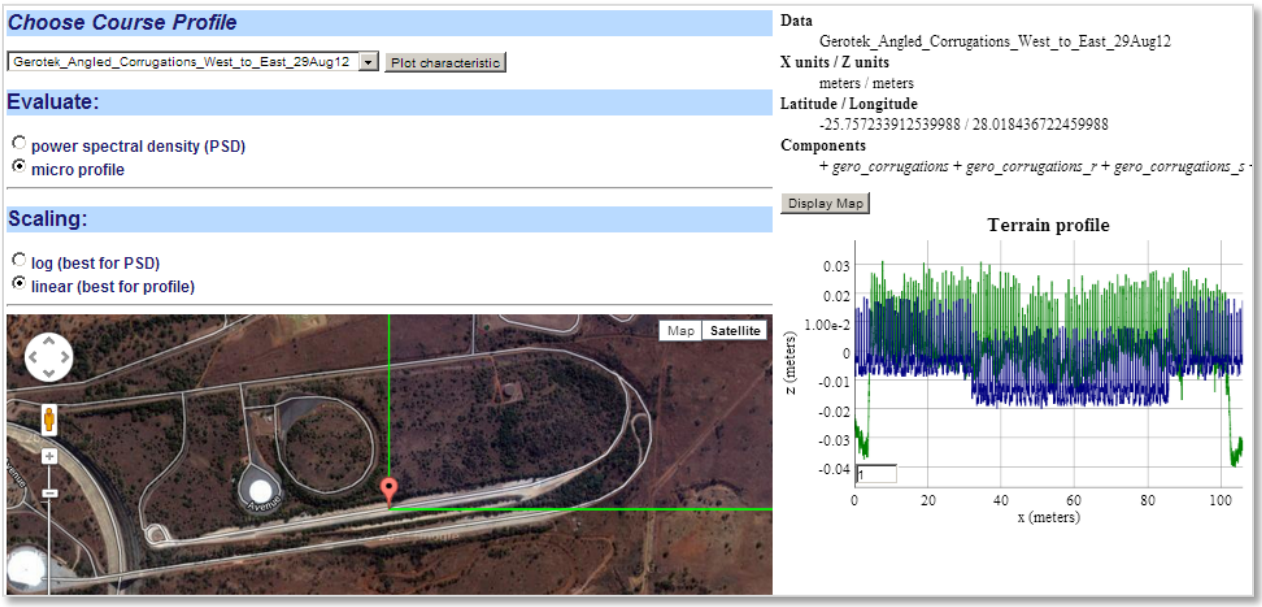


Figure 40 *: The typical fitting procedure uses knowledge of the underlying terrain profile (top middle inset, showing a pot-hole course with steps) and a stochastic representation of the PSD (upper right inset) to match the PSD calculated from the data.*

We essentially use the PSD to estimate the strength of the periodicity in the underlying terrain features. The sharper and stronger the harmonic peaks are, the stronger the periodicity.

For example, **Figure 41** shows the terrain and model for a corrugated course profile which shows strong periodic features.



↓ **Map zoomed below**

Figure 41: *Workflow for Gerotek corrugated course. The terrain profile is indexed to a Google Map of that area and we can zoom in and pick up the scale of the highlighted corrugated features.*

The following models (**Figure 42**) show a range in periodicities ranging from very strong (corrugated course), to easily detectable but disordered (Belgian block cobblestone course), to weak (random spaced obstacles).

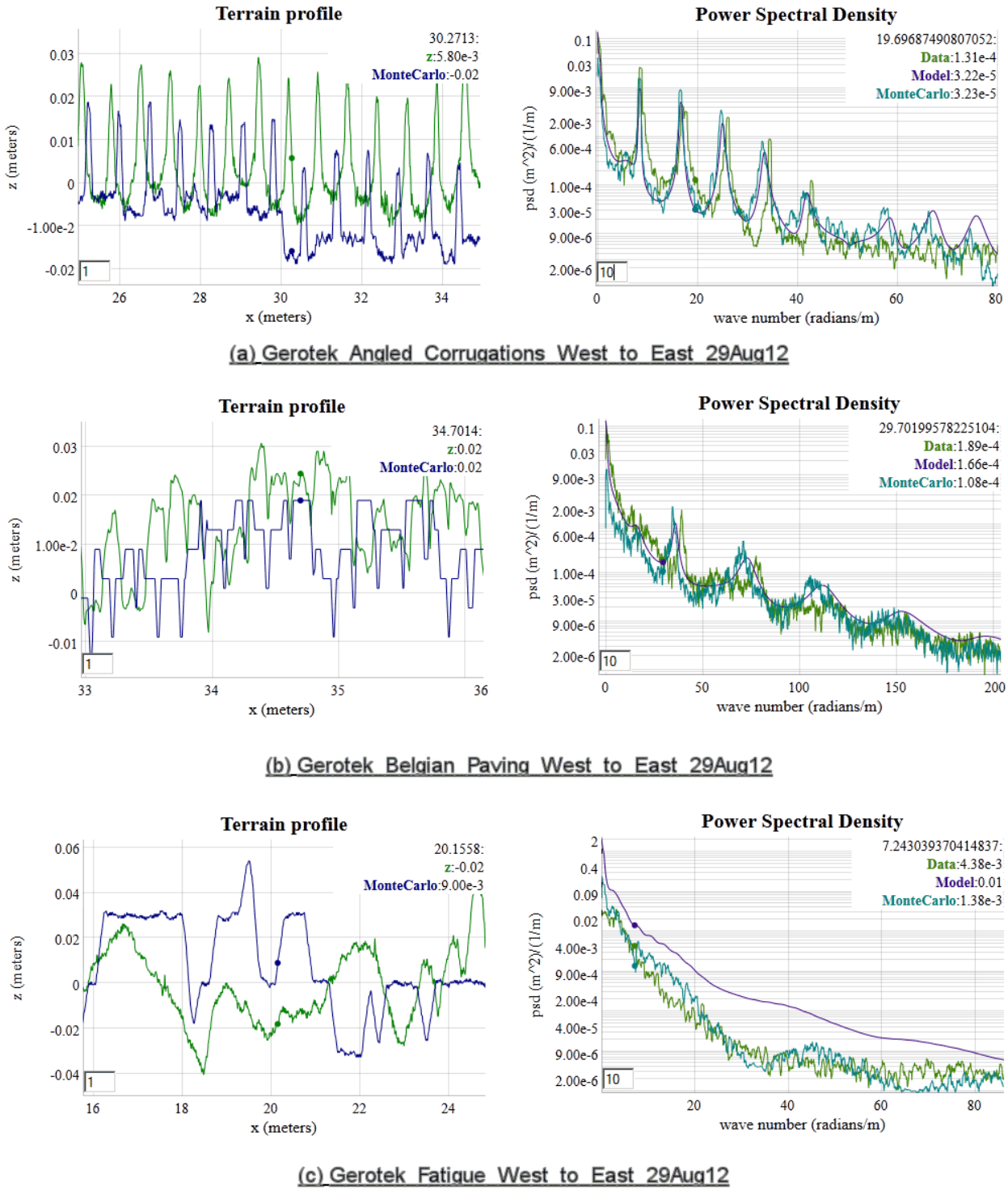


Figure 42: *Semi-Markov model fits for high-resolution Gerotek courrses. For each terrain, a profile of the actual terrain is shown as the bright green, and the model as the blur. Course (a) is a fairly reguler layout of square bumps. The missing harmonic at 50 radians/m is due to the width of the bump. Course (b) is a worn layout of Belgian block cobblestones, with periodicity emerging from the block spacing according to the PSD. Course (c) is a course with random obstacles and very weak periodicty.*

Superposition of Sine approach

This is essentially a Fourier series reconstruction approach based on having only empirical PSD data available.

For each course, identifying characteristics are provided along with a PSD plot of the course profile. The plot is usually attached to an otherwise classified course layout (see **Figure 43**).

Figure 43: *TOPS test course*

The figure below shows the workflow for selecting from PSD course data (see **Figure 44**).This indexes against an ontologically categorized set of terrain PSD data cataloged from Test Operating Procedures (TOPS)[20]. The data set contains header information necessary to calibrate the scales.

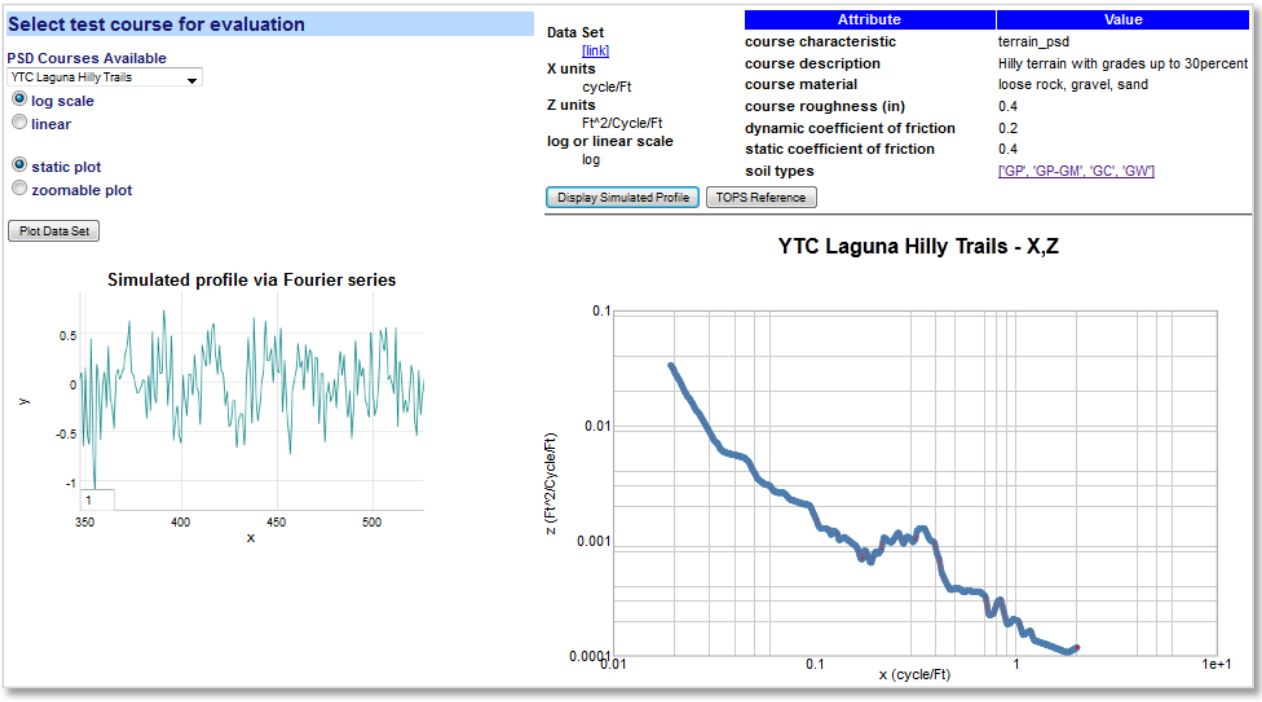


Figure 44: *PSD workflow example. The PSD data set is selected from a drop-down list. The PSD is plotted on the lower right and a Fourier series reconstruction is displayed on the lower left. If soil types are available, they are linked to a soil classification table.*

Gross-Relief Topographic View

Terrain Elevation models marginal probabilities shown on a log scale indicating likelihood of an elevation change given a surface translation.

Terrain is characterized for the lower 48 states according to digital elevation model (DEM) [38][39][40] profiles. The surface translation is limited to 40 post-spacing's, which corresponds to a range of less than 4000 meters and the elevation change is limited to 20 meters.

Fits for each 1 degree section are generated via a Gauss-Markov model following an Ornstein-Uhlenbeck (O-U) reversion-to-the-mean random walk process. Maximum entropy (MaxEnt) estimation generates uncertainty in the O-U model, which in general gives a better fit to the terrain data sets. (see further Appendix F for an application). For a selected point on the map (**Figure 45**) or the drop-down list of a named section (**Figure 46**), the enclosing 1º DEM quadrangle is analyzed for elevation correlations.

Figure 45: *Map selection mode pick a 1 degree quadrangle, shown by the green outline on the right.*

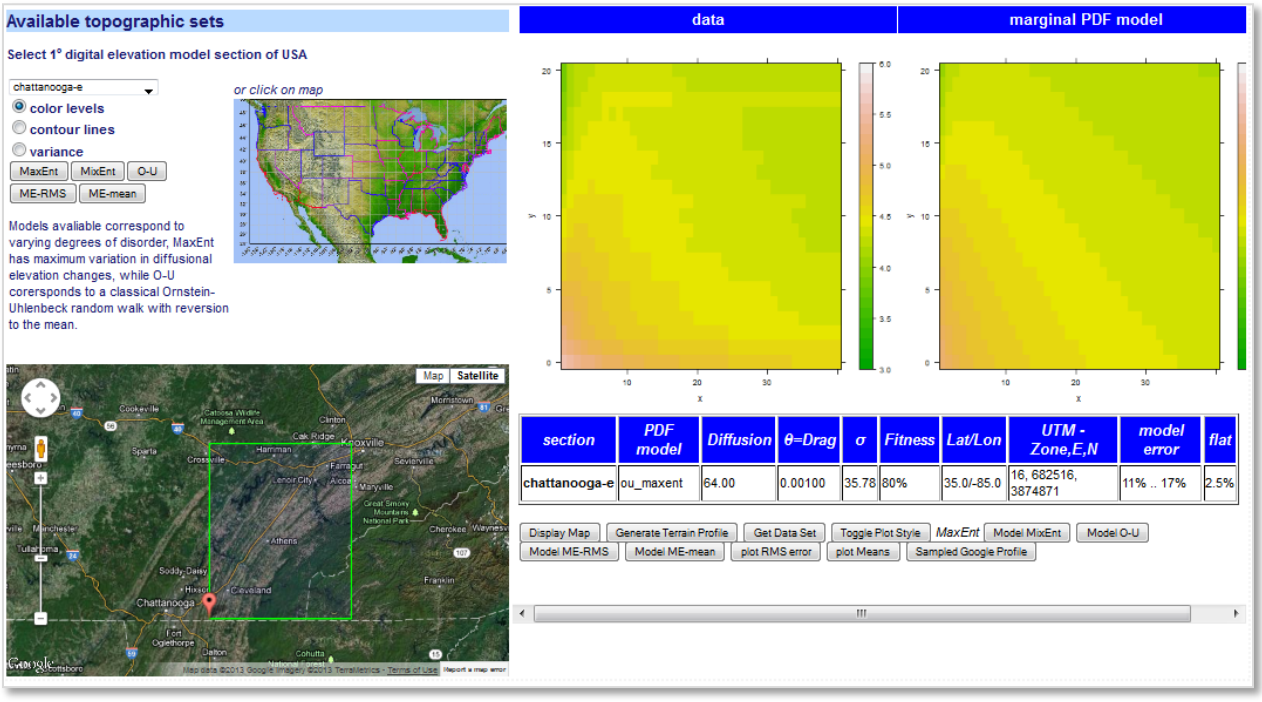


Figure 46: *Elevation correlations are shown as color contour maps. The model fit is shown in the far right, which should match in levels to the data contour next to it. This is a correlation for the Chattanooga east DEM, shown in the lower left.*

As the color contours may be difficult to delineate, a conventional line contour display is available as shown in **Figure 47**. Both types of contours were rendered via calls to the **R** statistics package.

If the model fit for a specific region matches the contour profile with relatively small error, then the model can be used to simulate a typical terrain elevation profile for that section. So the model essentially takes the place of actual contours from the region; for example **Figure 48** (a) is the model and **Figure 48** (b) is a randomly located sample from a Google Map query in that same section.

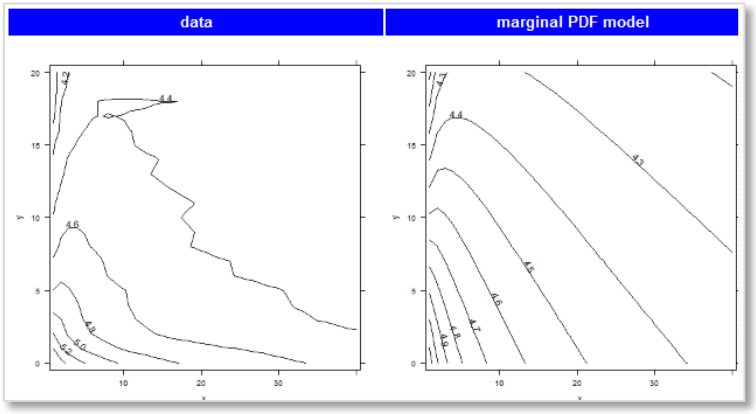


Figure 47: *Contour plot variation*

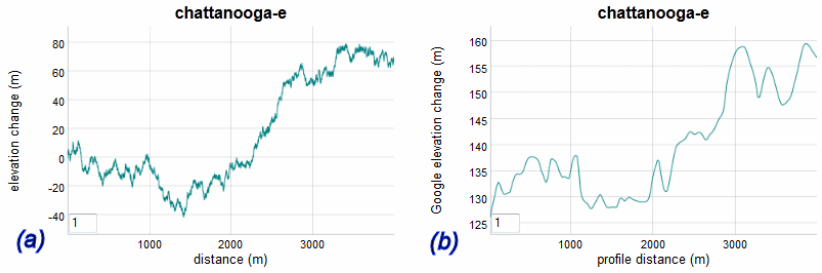


Figure 48*: The fit to the DEM section via an Ornstein-Uhlenbeck model is translated to a random walk profile of terrain in (a). A sample profile extracted from a Google Map API query is shown in (b).*

Temperature

The characterization of temperature is obviously acutely dependent on geospatial location of the simulation, test, or use case. Although environmental standards handbooks provide nominal characteristics for cold and hot climates [31] (which we cover elsewhere in the ontology), the need often arises for providing temperature models for specific locales.

Seasonal Model

We leveraged linked semantic data from other open sources to demonstrate what is possible with the context server. The site **dbpedia.org** [41] provides average monthly climate data as part of their instance database, and other sites such as Wikipedia already rely on it. For this example, we demonstrate how it is integrated with the dynamic context server.

The server essentially takes two steps in its workflow reasoning. It first requests all available locations with temperature from **dbpedia**, and then once the user selects an available location, the monthly high and low averages are plotted (see **Figure 49**).

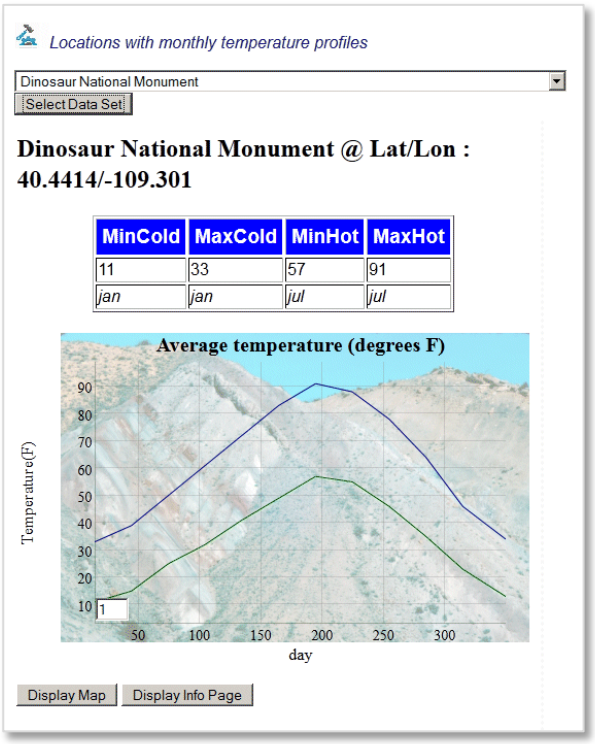


Figure 49: *Average monthly high and low temperature extremes captured from* ***dbpedia****. The background image is also supplied as a linked data element to the geospatial location.*

Other atmospheric measures are also available depending on whether a specific location (and dbpedia) has the data recorded. In **Figure 50** below, we show average precipitation for a semi-arid location in Oregon.

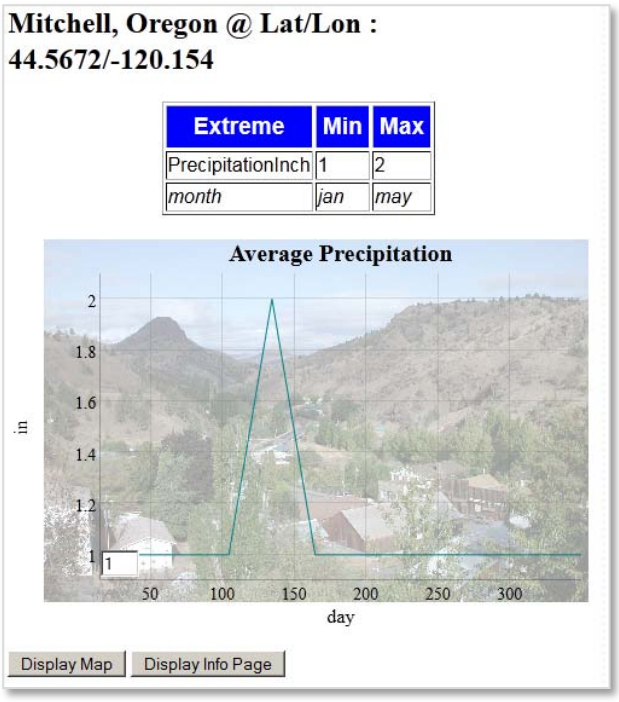


Figure 50: *Average monthly precipitation of eastern Oregon acquired from* ***dbpedia***

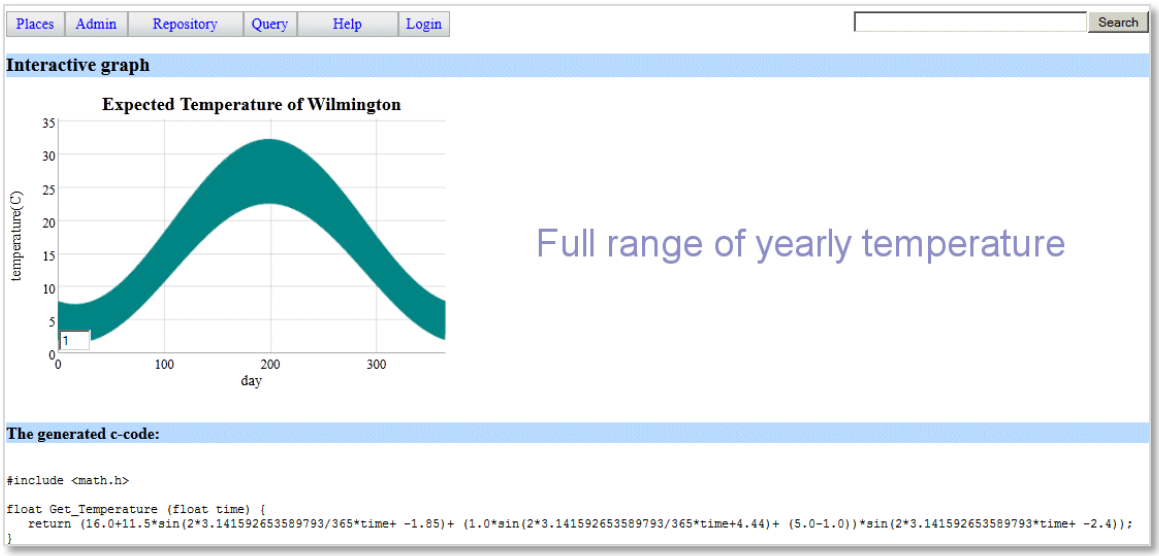
Diurnal precision model

As part of the AVM project, we learned that context model users requested more temporally-detailed temperature records than were available from **dbpedia**. In particular, diurnal records were requested for Camp Lejeune in North Carolina and also for Aberdeen Proving Ground near Baltimore, Md.

For North Carolina, the closest sites we could find in NOAA national weather service records were the Wilmington and Morehead City sites. These are equally close to Camp Lejeune according to the NWS records (http://www.erh.noaa.gov/). This situation points to the hit-and-miss nature of raw non-semantically indexed records. The historical data at the Wilmington location was in a PDF file, so we had to extract the temperature data manually. The Morehead City page had a slot but no data. Baltimore had a text file which required custom parsing. This is referred to as "unique local data", and the culling of this data for semantically-aware context usage is part of the knowledge capture process, and which shows why the structured approach of **dbpedia** is so effective.

After extracting the data from the National Weather Service records for these two locations, expected value models were generated to give the "normal" temperature that a weather forecaster would provide for any calendar date and time-of-day (no statistical moments were generated). **Figure 51** shows the diurnal model over a calendar year. The higher resolution mode of the dynamic graph functions by dragging the cursor across a calendar range and then the daily periodic nature becomes obvious.

Some of the non-uniformity in reporting is being addressed by NOAA, and they have recently deployed the U.S. Climate Reference Network to streamline the reporting: http://www.ncdc.noaa.gov/crn/



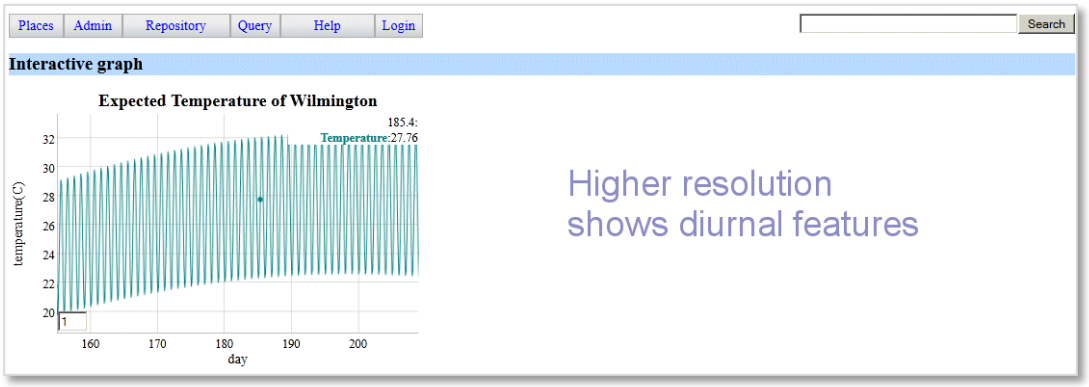


Figure 51: *Diurnal temperature model for Wilmington, NC.*

As the functional form of the expected diurnal temperature profile is saved in symbolic format, the semantic reasoner can convert that to a source code output. Below is a C-code snippet that uses the triple-store parameterization for a specific location.

```
#include <math.h>

float Get_Temperature (float time) {
   return (16.0+11.5*sin(2*3.141592653589793/365*time+ -1.85)+
(1.0*sin(2*3.141592653589793/365*time+4.44)+ (5.0-
1.0))*sin(2*3.141592653589793*time+ -2.4));
}
```

The input (time) is in days since the first of the year, so noon on January 1st would be input as 0.5. Noon on the last day of the year is 364.5.

The semantic architecture can potentially apply geospatial reasoning[42] to pull in appropriate data. Indeed, geospatial functionality has found widespread use for inferring locality information. For example, say that temperature statistics are not known for a particular area, but data from nearby locations is available. A geospatial engine can aggregate and select data from weather stations in close proximity and use that to interpolate the statistics.

Thermal Diffusion

Models of thermal transients follow the boundary conditions set by temperature constraints[4]. As temperature always seeks equilibrium, the boundary conditions will be compliant if large thermal masses added to the environment. This means that the thermal transient will likely asymptotically approach some intermediate value of temperature factored by the ratios of the thermal masses involved.

The model used here applies a thermal diffusion process with a range in uncertainty of the coefficient of diffusivity or conductance.

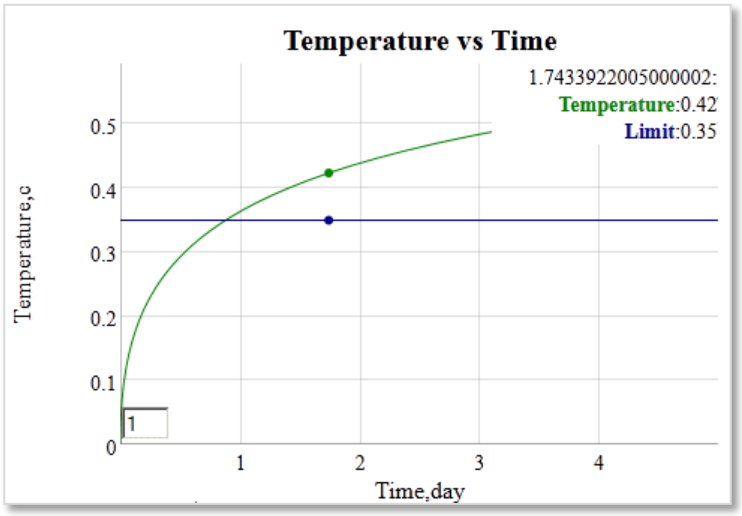


Figure 52: *Thermal dispersion model assuming uncertainty in the coefficient of thermal conductivity, and a unit step thermal stimulus.*

Stream Fording

Empirical information on stream flow suitable for evaluation of vehicle fording is subject to availability of discharge rates near dam sites. **Figure 53** shows a possible trajectory for an amphibious vehicle to cross a stream. Enough data is available that modeling of stream flow is possible[43].

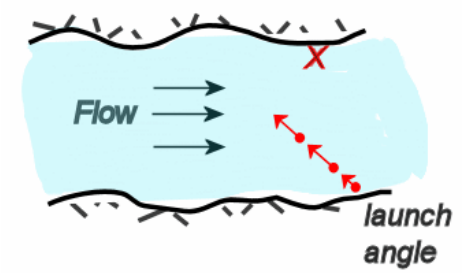


Figure 53 : *Planning trajectory for fording a stream. To reach a designated point (X) on the far side of a stream, a launch angle and velocity is selected to compensate for the stream flow rate.*

The stream flow is related to discharge rate via the cross-section of the river at the location of the measurement. Some sites have this available and will translate it directly to current or stream flow rate, yet this is not the rule. A typical PDF for stream flow is shown in **Figure 54**.

The other factor affecting stream flow is the seasonal variation caused by local weather patterns, precipitation, and upstream run-off or flood control.

The approach taken here is to provide a pattern which can be extended to allow for parametric analysis for specific locations. The range of variability in currents and discharge rates is great enough that a predictive generic model is not possible for the current statistical information.

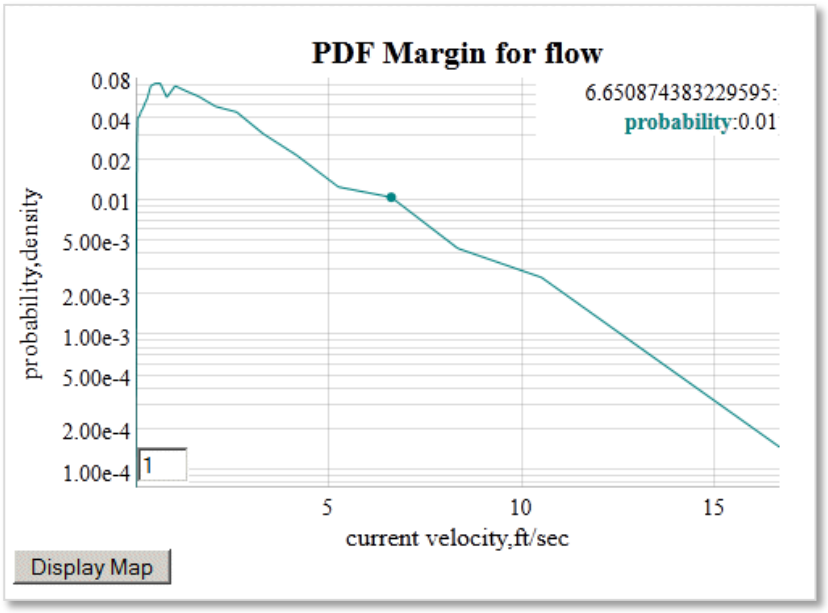


Figure 54: *PDF for marginal flow of the Allegheny river. To convert from discharge rate to current velocity, a river cross-section (width × depth) was assumed.*

Clutter Integration

A clutter example demonstrates the time it takes to integrate an electronic signal buried in environmental EMI noise. [2]

In this case, a GPS is initiated from a cold-start and the probability that the GPS is locked on a position after a specified time is shown in **Figure 55**.

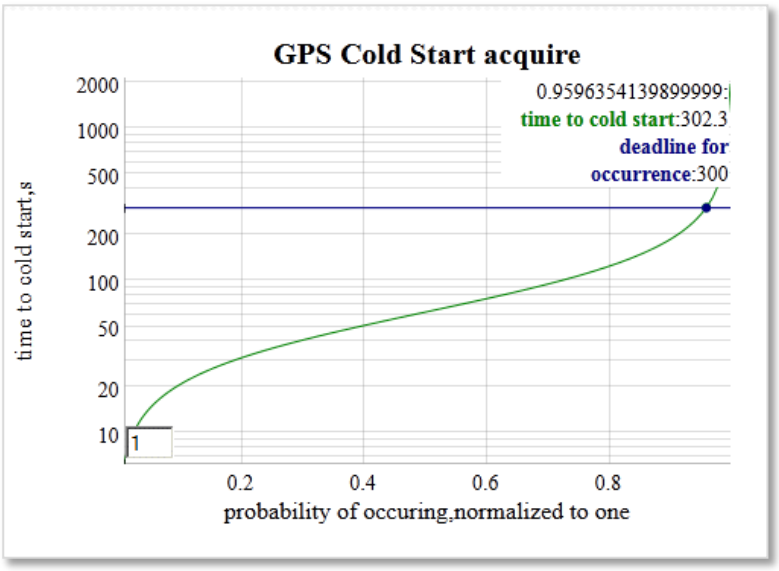


Figure 55: *Time it takes to lock into a GPS signal from a "cold-start". This integrates the statistical EMI noise until a certain level is reached. The result at the deadline is a probability of lock-in.*

Corrosion Example

Corrosion of metals is a slow process, accelerated by adverse environmental factors such as high levels of salt and acidity.

	Marine(severe)	Industrial	Marine(mild)	Urban	Rural
D	90,000	45,000	12,000	6,800	5,300
X_0	600	400	100	50	500
a	1.00E-08	1.00E-08	1.00E-08	0.4	0.2

$$X(t) = \sqrt{\frac{D \cdot (1 - e^{-at})}{a}} \cdot \left[\frac{\sqrt{\frac{D \cdot (1 - e^{-at})}{a\, X_0}}}{1 + \sqrt{\frac{D \cdot (1 - e^{-at})}{a X_0}}} \right]$$

We apply a diffusional model of corrosion which emulates the temporal profile of the empirical observations while providing an intuitive understanding to the gradual processes involved.[4]

The empirical parameters were taken from a recent study which considered long term corrosive effects in variously categorized environments. **Figure 56**

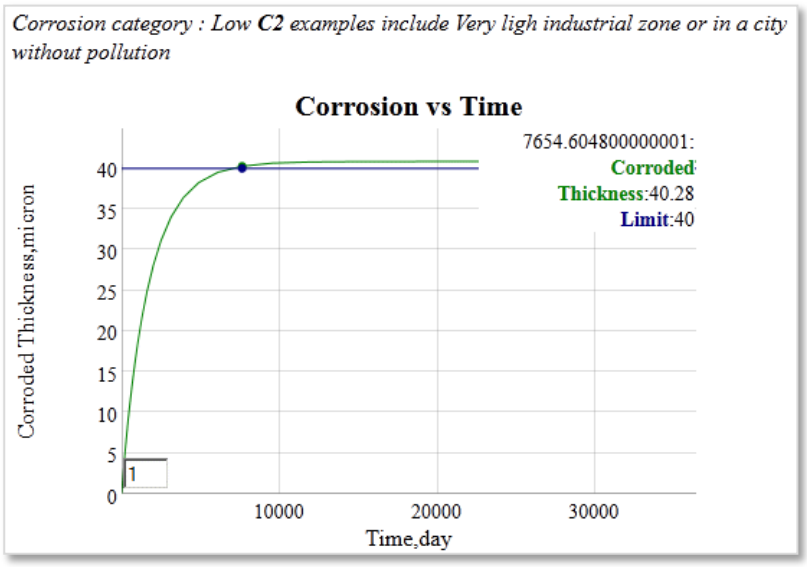


Figure 56: *Example of a corrosive growth model parameterized from a low pollution site.*

Droplet Size

For particulates, we consider a standard specification of rainfall droplet size from an environmental standard[31].

Table 2: *From AR 70-38 standard population density against raindrop size*

	Drop Diameter Range (mm)					
density	0.5 - 1.4	1.5 - 2.4	2.5 - 3.4	3.5 - 4.4	4.5 - 5.4	5.5 - 6.4
# / m^3	2626	342	45	6	1	<1

The histogram in **Table 2** is an interpolated fit of the drop diameter data to a PDF based on two parameters, a mean droplet size that follows an exponential decline, mixed with a much lower exponential decline probability of a droplet with a higher size. The mix generates the slight knee in the curve shown in **Figure 57**.

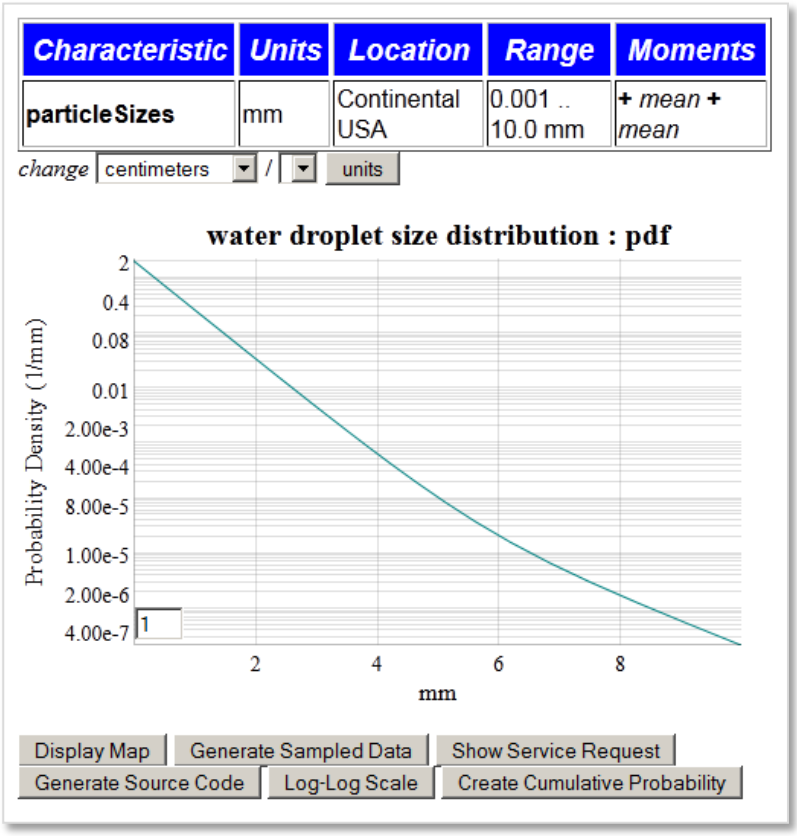


Figure 57: *PDF of water droplet size distribution*

This class of PDFs is indexed via the pattern described in Annex 2: PDF Models.

Wind

We captured the wind velocity distributions from several regions of the world as a set of parameterized PDFs. The characterized distributions were fit to maximum entropy models as described elsewhere[2].

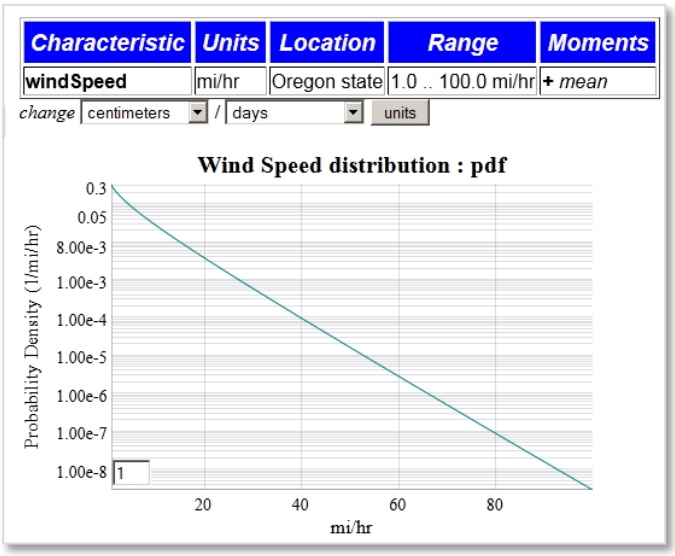


Figure 58: *Wind speed probability density function for data from the Bonneville Power Authority in Oregon.*

Figure 58 shows the result of choosing a certain region (Oregon's BPA data set) and then plotting the PDF from a BesselK wind speed distribution. The unit dimensionality rule was attached to allow the user to change the wind speed variate to any ratio of distance/time drawn from the triple-store unit definitions.

A sampling query is also available for drawing values from the PDF of the selected model. The BesselK requires two independent samples from an exponential damped distribution[2].

Pressure

A set of standard atmospheric profiles is encoded to allow a user to quickly estimate measures such as pressure at different altitudes. A pre-selector for altitude units and altitude specified is used as input to generate the graph shown in **Figure 59**.

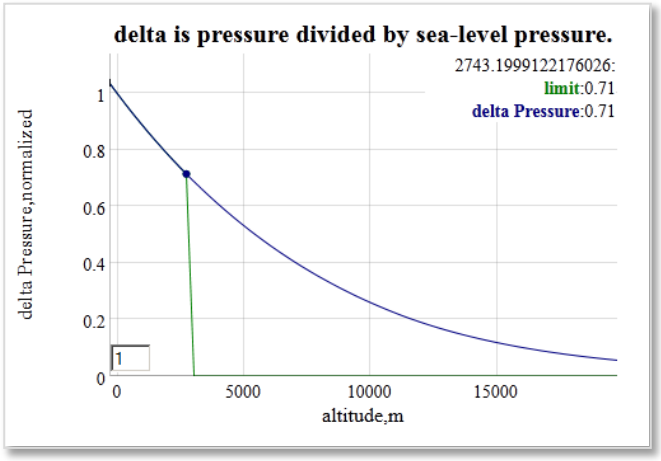


Figure 59: *Pressure for standard atmosphere as a function of altitude.*

Buoyancy Example

For aquatic environments we demonstrate a buoyancy model that compares density of immersed volumes against water density. We select a mass of an object and the volume that it displaces. To keep it obvious, we select an object that has about the same density of freshwater, 1 g/cm^3 or 1000 kg/ m^3. A rough prototype of the object density selection is shown below in **Figure 60**[3].

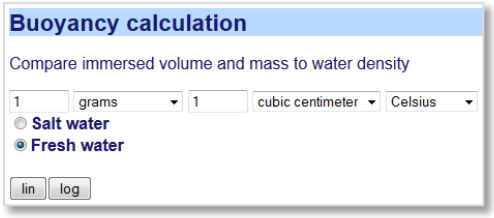


Figure 60: *Prototype for buoyancy input*

In the figures below, one can see that the object is critically buoyant near freezing temperature (**Figure 61**) but at higher temperatures (**Figure 62**), it becomes denser than water and therefore will sink.

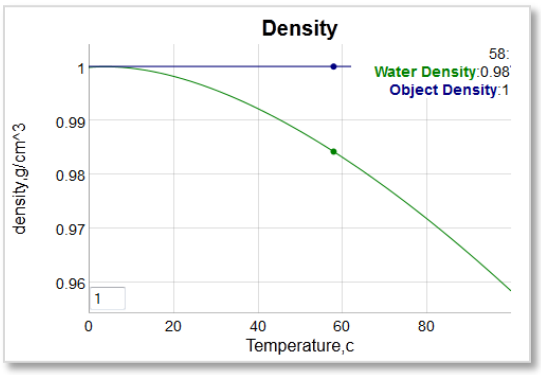


Figure 61: *Critical buoyancy calculation for fresh water. If the green water density line is above the blue object density line, the object is buoyant and will float. (note that units in g/cm^3).*

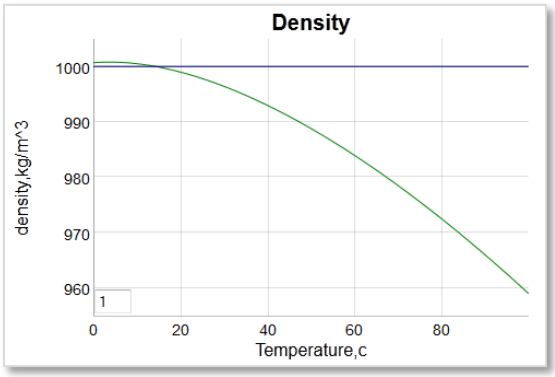


Figure 62: *For salt water, note the slightly higher density, so object is more buoyant. (alternative units in kg/m^3).*

[3] As the buoyancy example is limited, a link to a more detailed spreadsheet model is provided in the example page. This includes instructions for creating a ProE model for the immersed object, which can calculate immersed density and center of gravity for more complex structures. Here we concentrate only on the context.

Patterns of Usage

All the examples were created from rules operating on semantic triple stores. Many of the examples operated on a standard pattern; for example, the wind speed and rain droplet size used the same PDF generator (see Annex X for a complete list).

For general external use, triple stores that are exposed by the dynamic context server can be accessed via a query language such as SPARQL. For example, the SPARQL expression

```
select * where{?s relaSci:hasNumericValue ?o
```

can be transformed into a web service query by prefixing a "`sparql/?query=`" to the SPARQL expression. Then the following service call

```
http://localhost:3020/sparql/?query=select * where{?s relaSci:hasNumericValue ?o}
```

will return an XML list of all the triples that contain the predicate `:relaSci:hasNumericValue.`

So the examples of usage described in this section can get integrated with other semantically-aware languages that can either create web requests as URL expressions or as SPARQL queries. Having some facility with how to read triple-store graphs and construct URLs, a savvy developer can reuse the web services and/or browse through triple-store graphs

```
http://localhost:3020/browse/list_graphs
```

Development Process

The development of the semantic web software was facilitated by several collaboration tools. As this was part of the AVM effort, we took advantage of the collaboration tools supplied to the project teams shown in **Figure 63**. The Confluence Wiki provided the anchor for the semantic web development, as it provided a scratchpad for architectural ideas and for sharing knowledge and data for the context model library.

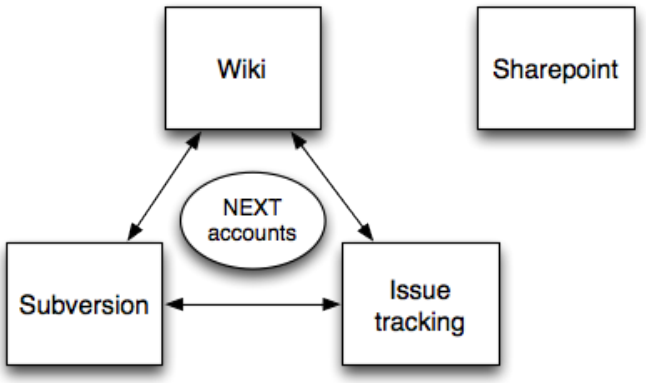


Figure 63: *Team collaboration tools used during the development process.*

Summary

The Dynamic Context Server provides models and artifacts as part of a comprehensive environmental modeling library. To maintain the library with the necessary level of organization and categorization, we used the Semantic Web for Earth and Environmental Terminology (SWEET) ontology as a semantic and terminology base. The system was built as an interactive web server that consisted of the DCS with an adjunct model delivery service called Ontological System for Context Artifacts and Resources (OSCAR). OSCAR's capabilities do not overlap with the DCS capabilities and in fact the two complement each other, with the two views integrated through a conventional web-service front-end.

The physical domain models represent the land, atmosphere, and aquatic realms. These have deterministic and stochastic representations; as an example from the terrain realm, the stochastic terrains characterized by PSD's and Markov models, and the obstacles as deterministic fixed geometries.

As architected, DCS and OSCAR perform as portals for serving context models and knowledge which can meet the needs of vehicle design and test.

Annexes

Annex 1: Browser Narrative

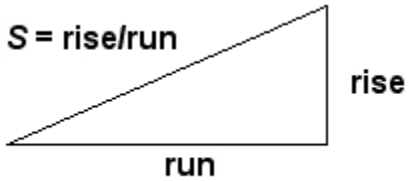
The server supplies an online narrative that describes the library hierarchy of major environmental classifications, **Land, Aquatic, and Atmospheric.** It is reproduced below.

The **Land** classification consists of all terrestrial models, specifically those attached to solid earth. This includes both natural terrain and man-made obstacles, and anything that deals with the terrestrial surface. Natural terrain is largely stochastic, with elements of determinism mixed in via certain man-assisted or physical processes.

Terrain is defined by the vertical and horizontal dimension of land surface. Characteristics of the terrain is usually expressed in terms of the elevation, slope, and orientation of terrain features

The **gross terrain** is also referred to as topography or land relief. The deterministic view of terrain is typically represented by a scaled elevation contour plot of a specific region of land. The stochastic view is represented by models of the topography, which is often cast in terms of a random walk process.

The **slope** of the terrain is characterized by the empirically measured rise/run of the local surface area. It is also known as the grade or pitch. By orientating perpendicular to the maximum slope, a pitch characterization becomes a roll characterization. In general, natural terrains show greater variation in slopes than do man-made features, such as roads or rail-road tracks, which will show signs of grading and switchbacks or other grade limitations.



A characterization of terrain **elevations** is usually correlated against a geospatial surface dimension. Analyzing pair correlations between elevations separated by a lateral surface dimension allow models of regional topography to be made. By using stochastic models of the terrain with well-characterized probability distributions, one can use that as an input or constraining stimulus for navigability and formal verification.

Characterization of **fine terrain** requires a scale much less than the topography or land relief of the local region. To remove this macroscopic effect of large scale elevation and slope changes, data is often detrended to reveal only the fine detail.

The fine terrain **roughness** is characterized by a pair correlation function and the frequency representation known as the power spectral density. The typical terrain roughness may follow a random enough pattern so that the information contained in a correlation function or PSD is enough to reveal the stochastic nature of the ground.

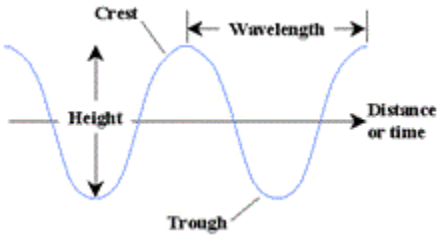
Oftentimes the fine terrain **profile** shows enough regularity that the addition of non-randomness to the model becomes effective in reproducing the spectral properties. Fine terrain with elements of randomness can include washboard surface and cobblestone roads. Individual profiles are often referenced for further evaluation and usually referred to as test tracks or courses when used in the context of vehicle evaluation. A stochastic model of the terrain, if it is characterized fully, can reduce the storage data requirements by orders of magnitude, leaving a few parameters to describe a specific course profile.

Discrete **obstacles** are always characterized by a geometrically defined spatial profile.

Soil classification schemes simplify the cataloguing of the variety of naturally occurring soils into a handful of types according to a few properties such as granularity.

The **aquatic** classification consists of all ocean and inland water models, and specifically those dealing with the surface characterization of a particular body of water. This can include wave height, lake sizes, and river current speed.

The **sea state** of the water surface at a specific location and time qualifies its general conditions by categorizing according to wave height and other related characteristics such as period and power spectrum. Higher sea-states correspond to rougher conditions, with zero indicating calm waters.



By empirically measuring the historical **wave height** in a specific location and attaching probabilities to the occurrence of various wave heights, we can generate the probability of encountering a particular sea state.

The **wave frequency** is closely related to wave height and wave frequency by a dispersion relationship. Because of the common hydrodynamic properties of water, it is often enough to consider only wave heights when inferring the frequency and wavelength of water.

The distribution of **lake sizes** and **stream currents** has a stochastic explanation, and models of these distributions can fit to empirically collected statistics.

Characterizing **water density** allows for the effects of buoyancy to be determined. Both temperature and salinity can subtly affect the density of water.

The category of **atmospheric** models contains phenomenon that relate insofar as they propagate through the air.

The variation of **wind speed** in specific geographic locations is predominantly a stochastic phenomenon. The general trend is of decreasing likelihood of wind speeds with increasing magnitude.

Precipitation in the form of rain and snow varies according to geographic location and season. Models of rainfall are available, as are environmental standards which give nominal conditions for certain regions and climates.

The statistics of **rainfall amount** has a foundation in extreme value theory, yet the commonly occurring measured rainfalls follow distribution functions that can be modeled.

Based on recently accumulated data, the areal extent of **clouds** has been shown to follow a simple stochastic model.

The modeling of atmospheric **temperature** has a lower spread in relative uncertainty than other environmental contexts. This has much to do with the large thermal inertia of the planet and of the predictably of diurnal and seasonal change. Thus, models of temperature are better suited to description by a mean value and small *relative* (on the Kelvin scale)

The model of **solar** insolation is deterministically predictable, abated only by sporadic cloud cover.

The diurnal or **daily and seasonal** temperature variations can be captured for specific geospatial locations and modeled over the course of a full year, including a moving average for the 24-hour period. In terms of nominal conditions, representative regions can be defined as "hot" to "extreme cold" and typical mean values can be generated to cover the expected values.

Models of transient response to **thermal** steady-state require compliance between the system under study and the environment. In general terms, this means that the system can influence the environment enough to affect the dynamics and the eventual steady-state point (if one exists).

The **humidity** of a particular region is given by nominal conditions. Since humidity only plays a contextual role over a long term, in say promoting corrosion, the nominal values are adequate. Therefore, environmental standards serve as better defining documents than models based on empirical data for a specific location.

Airborne **particulates** can show persistent or transient effects. After significant natural events such as the eruption of volcanoes, the density and size of particulates can increase rapidly. Otherwise, a heterogeneous mix of particles from different origins describes the typical distribution.

Airborne particulates can thus show great diversity in **particle size**. If the particles can grow over time, such as ice crystals, the steady-state distribution in parts reflects where the particle is in with respect to its life cycle. The same process occurs with suspended water-borne particulates.

Airborne particulates show ranges in **particle density** usually depending on the transient event.

Background noise from the environment is referred to as **clutter**

Radio and radar transmitters can present very high levels of energy and electromagnetic field strength to a system and the equipment and subsystems it contains. This type of environment is referred to as the external **Radio Frequency (RF) Electromagnetic Environment (EME)**. RF EME is described in terms of field strength as a function of frequency.

An **electromagnetic pulse** is a rare occurrence of either natural or man-made origin. One variation of this is high power high power microwave which can be a single pulse or transmitted as a repetitive waveform.

Lightning effects can be either direct or indirect. The direct effects refers to the situation were the lightning channel couples to the system, whereas indirect refers to situations where the lightning is observed only remotely through sound, flashes, vibration, or interference.

Static electricity can interact with the environment to create an **electrostatic discharge**

A vehicle can also generate **internal sources of EM energy**, which by way of proximity can interfere with objects in the environment.

Annex 2: PDF Models

The set of PDF models follow a pattern of a variate with parameters that allow us to collect all of the information as triple-stores.

```
PDF mapdot K*f(A,B,C) ~> X
```

What this does is apply a lambda function with scaling factor K and fixed parameters A,B,C to a variate list X ranging from Min to Max (with a step interval). The entire list is shown in **Figure 64**.

Characteristic	Model	Title	Xaxis	Location	Lat/Lon	PDF	Moments	Max	Min	Interval
cloudArea	Ocean_clouds	cloud diameter distribution	m	Pacific near Hawaii	17.5/-162.5	1.0*power_law_2(0.1,Dist)	[median]	100000.0	1.0	1.1
clutterPower	Variable_clutter	clutter power distribution Rayleigh	w/m^2	The World	0.0/0.0	1.0*exp(1.0,Dist)	[mean]	10.0	0.01	1.1
clutterPower	Variable_clutter_maxent	clutter power distribution MaxEnt	w/m^2	The World	0.0/0.0	1.0*besselk0_sqrt(1.0,Dist)	[mean]	10.0	0.01	1.1
lakeSize	Amazon_lakes	lake size distribution	km^2	Amazon region	-5.0/-60.0	1.0*power_law_2(0.19,Dist)	[median]	1000.0	0.01	1.05
lakeSize	Canadian_lakes	lake size distribution	km^2	Northern Quebec	54.0/-75.0	1.0*power_law_2(0.1,Dist)	[median]	1000.0	0.1	1.05
particleSizes	ice_particles	ice particle size distribution	micron	Continental USA	38.0/-100.0	[0.995*power_law_2_area(3.0,Dist), 0.005*exp_area(3000.0,Dist)]	[[median], [mean]]	10000.0	1.0	1.1
particleSizes	particles	particle size distribution	nm	A Volcanic area	63.0/-16.0	[0.5*exp(3.0,Dist), 0.5*exp(0.1,Dist)]	[[mean], [mean]]	10.0	0.001	1.1
particleSizes	rain_drops_standard	water droplet size distribution	mm	Continental USA	38.0/-100.0	[0.9951*exp(0.489,Dist), 0.0049*exp(0.996,Dist)]	[[mean], [mean]]	10.0	0.001	1.1
rainfall	Iowa_rainfall	Rainfall Distribution	mm/hr	Iowa state	41.658/-91.548	1.0*besselk0_sqrt(2.29,Dist)	[mean]	100.0	0.001	1.05
rateNumber	lightning_rate	lightning rate distribution	num/min	Continental USA	38.0/-100.0	1.0*diffusion_accel(16,5.5,Dist)	[[diff], [accel]]	100.0	0.1	1.1
slopes	CONUS_slopes	Slope Distribution	rise/run	Continental USA	38.0/-100.0	1.0*besselk0_sqrt(0.037,Dist)	[mean]	10.0	0.01	1.05
waveFrequency	San_Diego_waves	Wave frequency Distribution	hz	Point Loma South	32.67/-117.2419	1.0*pierson_moskowitz(0.085,Dist)	[mean]	0.6	0.05	1.01
waveFrequency	San_Diego_waves	Wave frequency Distribution	hz	San Nicolas Island North	32.25/-119.5	1.0*pierson_moskowitz(0.085,Dist)	[mean]	0.6	0.05	1.01
waveHeight	Atlantic_seaboard_waves	Wave height Distribution	m	Atlantic area	35.0/-75.0	1.0*bessel_seastate(1,0.81,26.0,Dist)	[mean, depth]	16.0	0.001	1.05
waveHeight	Michigan_waves	Wave height Distribution	m	Lake Michigan	43.0/-87.0	1.0*bessel_seastate(0,0.56,12.5,Dist)	[mean, depth]	10.0	0.001	1.05
waveHeight	Superior_waves	Wave height Distribution	m	Lake Superior	49.0/-90.0	1.0*bessel_seastate(0,0.85,16.0,Dist)	[mean, depth]	12.0	0.001	1.05
windSpeed	Germany_wind_energy	Wind Energy distribution	MW-Hr	Germany	49.0/9.0	1.0*exp(1737,Dist)	[mean]	10000.0	1.0	1.05
windSpeed	Ontario_wind_energy	Wind Energy	MW-Hr	Ontario	52.0/-82.0	1.0*exp(178,Dist)	[mean]	1000.0	1.0	1.05

		distribution								
windSpeed	Oregon_wind_energy	Wind Speed as Energy distribution	MPH^2	Oregon state	43.0/-121.0	1.0*besselk0_sqrt(144,Dist)	[mean]	10000.0	1.0	1.1
windSpeed	Oregon_wind_speed	Wind Speed distribution	mi/hr	Oregon state	43.0/-121.0	1.0*besselk0(12,Dist)	[mean]	100.0	1.0	1.05

Characteristic	Model	Title	Xaxis	Location	Lat/Lon	PDF	Moments	Max	Min	Interval
cloudArea	Ocean_clouds	cloud diameter distribution	m	Pacific near Hawaii	17.5/-162.5	1.0*power_law_2(0.1,Dist)	[median]	100000.0	1.0	1.1
clutterPower	Variable_clutter	clutter power distribution Rayleigh	w/m^2	The World	0.0/0.0	1.0*exp(1.0,Dist)	[mean]	10.0	0.01	1.1
clutterPower	Variable_clutter_maxent	clutter power distribution MaxEnt	w/m^2	The World	0.0/0.0	1.0*besselk0_sqrt(1.0,Dist)	[mean]	10.0	0.01	1.1
lakeSize	Amazon_lakes	lake size distribution	km^2	Amazon region	-5.0/-60.0	1.0*power_law_2(0.19,Dist)	[median]	1000.0	0.01	1.05
lakeSize	Canadian_lakes	lake size distribution	km^2	Northern Quebec	54.0/-75.0	1.0*power_law_2(0.1,Dist)	[median]	1000.0	0.1	1.05
particleSizes	ice_particles	ice particle size distribution	micron	Continental USA	38.0/-100.0	[0.995*power_law_2_area(3.0,Dist), 0.005*exp_area(3000.0,Dist)]	[[median], [mean]]	10000.0	1.0	1.1
particleSizes	particles	particle size distribution	nm	A Volcanic area	63.0/-16.0	[0.5*exp(3.0,Dist), 0.5*exp(0.1,Dist)]	[[mean], [mean]]	10.0	0.001	1.1
particleSizes	rain_drops_standard	water droplet size distribution	mm	Continental USA	38.0/-100.0	[0.9951*exp(0.489,Dist), 0.0049*exp(0.996,Dist)]	[[mean], [mean]]	10.0	0.001	1.1
rainfall	Iowa_rainfall	Rainfall Distribution	mm/hr	Iowa state	41.658/-91.548	1.0*besselk0_sqrt(2.29,Dist)	[mean]	100.0	0.001	1.05
rateNumber	lightning_rate	lightning rate distribution	num/min	Continental USA	38.0/-100.0	1.0*diffusion_accel(16,5.5,Dist)	[[diff], [accel]]	100.0	0.1	1.1
slopes	CONUS_slopes	Slope Distribution	rise/run	Continental USA	38.0/-100.0	1.0*besselk0_sqrt(0.037,Dist)	[mean]	10.0	0.01	1.05
waveFrequency	San_Diego_waves	Wave frequency Distribution	hz	Point Loma South	32.67/-117.2419	1.0*pierson_moskowitz(0.085,Dist)	[mean]	0.6	0.05	1.01
waveFrequency	San_Diego_waves	Wave frequency Distribution	hz	San Nicolas Island North	32.25/-119.5	1.0*pierson_moskowitz(0.085,Dist)	[mean]	0.6	0.05	1.01
waveHeight	Atlantic_seaboard_waves	Wave height Distribution	m	Atlantic area	35.0/-75.0	1.0*bessel_seastate(1,0.81,26.0,Dist)	[mean, depth]	16.0	0.001	1.05
waveHeight	Michigan_waves	Wave height Distribution	m	Lake Michigan	43.0/-87.0	1.0*bessel_seastate(0,0.56,12.5,Dist)	[mean, depth]	10.0	0.001	1.05
waveHeight	Superior_waves	Wave height Distribution	m	Lake Superior	49.0/-90.0	1.0*bessel_seastate(0,0.85,16.0,Dist)	[mean, depth]	12.0	0.001	1.05
windSpeed	Germany_wind_energy	Wind Energy distribution	MW-Hr	Germany	49.0/9.0	1.0*exp(1737,Dist)	[mean]	10000.0	1.0	1.05
windSpeed	Ontario_wind_energy	Wind Energy distribution	MW-Hr	Ontario	52.0/-82.0	1.0*exp(178,Dist)	[mean]	1000.0	1.0	1.05
windSpeed	Oregon_wind_energy	Wind Speed as Energy distribution	MPH^2	Oregon state	43.0/-121.0	1.0*besselk0_sqrt(144,Dist)	[mean]	10000.0	1.0	1.1
windSpeed	Oregon_wind_speed	Wind Speed distribution	mi/hr	Oregon state	43.0/-121.0	1.0*besselk0(12,Dist)	[mean]	100.0	1.0	1.05

Figure 64: *PDF models*

Annex 3 : Installation

The context server requires the installation of several run-times in addition to the DCS source.

Run-time Packages Required

1. SWI Prolog for Linux, Mac, or Windows (version w64pl636 for 64-bit Windows) http://www.swi-prolog.org/Download.html
2. R Statistics package (version 2.15.2) http://www.r-project.org/
3. AT&T Graphviz graphics package (version 2.28.0) http://www.graphviz.org/
4. Firefox preferred, but Google Chrome adequate as a browser. Only recent versions of Internet Explorer work well.

Make sure to install all the add-on packages for SWI if installation is not automatically installed. The Prolog runtime will need to know the path to R and Graphviz (this is automatic with a Windows install).

After extracting the source from the archive, go to **Ontology\dcs\dynamic_context_server** directory to start the DCS.

- **run.pl** (Windows registered)
- **run.sh** (Linux command line)
- **nohup-run-cloud** (running on a Cloud server as a "no-hang-up" job)

The port is 3020 for local clients and 80 for cloud configured (modify network.pl). If you have the DCS set up on a local machine then go to the following link: **http://localhost:3020**

An authorization login is required to administratively load the ontological context data, via menu item **Repository/Load Context Data**. This can be reset by modifying the names and MD5 hash passwords in the **users.db** file in the main **dynamic_context_server** directory.

Annex 4 : Units

We have collected several standard properties and constants as categorized triple-stores.

Standard Atmosphere properties

Attribute	Value
Name	standardAtmosphere
Comment	standard temperature and pressure (STP)
Description	Specification of standard atmosphere
dryAdiabaticLapseRate	9.8*c/km
dryAdiabaticPressureHead	-0.116*km
moistAdiabaticLapseRateTypical	5.0*c/km
molecularWeight	28.966*au
Pressure	1.0*atm
seaLevel	0.0*km
specificGasConstantDryAir	287*j/kg/k
specificGasConstantWaterVapor	462*j/kg/k
specificHeatRatio	1.4
Temperature	70*f
temperature	273.15*k

Water properties

Attribute	Value
name	water
comment	fresh water properties
deltaHvap	9717.1*cal
deltaSvap	26.04*cal/k
molecular_weight	18.015268*au

Solar

Attribute	Value
name	solar
comment	average sunshine
averageAlbedo	0.31
averageSolarInsolation	1366*w/m^2

Physical Constants

Attribute	Value
name	physicalConstants
comment	commonly used physical constants
atomicUnit	1.66053886e-24*g
avogadrosNumber	6.02214129e+23*n
boltzmannConstant	1.38065e-23*j/k
dielectricConstant	8.854187e-12*fd/m
electricalCharge	1.60218e-19*coulombs
faradaysConstant	96485.3383*coulombs/mol
gasConstant	8.314472*j/k/mol
gasConstant	1.9858775*cal/k/mol
gravity	9.80665*m/s^2
gravity	32.174049*ft/s^2
planckConstant	6.62607e-34*j*s
speedLight	300000000.0*m/s
stefanBoltzmann	5.670373e-8*w/m^2/k^4

Units suitable for conversion

Attribute	Value
All Units	symbolic
area	ft^2
area	m^2
density	g/cm^3
density	kg/dm^3
density	kg/m^3
density	lb/ft^3
density	mg/micron^3
density	oz/in^3
density	ton/km^3
dimensionless	percent
length	cm
length	ft
length	in
length	kft
length	km
length	m
length	micron
length	mi
length	mil
length	mm
length	nm
length	yd
mass	kg
mass	lb
pressure	atm
pressure	bar

pressure	millibar
pressure	mpa
pressure	mtorr
pressure	pa
pressure	psi
pressure	torr
temperature	c
temperature	f
temperature	k
temperature	r
time	day
time	decade
time	hr
time	mics
time	min
time	s
time	yr
volume	cm^3
volume	m^3

Annex 5 : Table of Acronyms

AC	Autocorrelation
ACF	Autocorrelation Function
APG	Aberdeen Proving Grounds
ATC	Aberdeen Test Center
BPA	Bonneville Power Administration
BesselK	Modified Bessel Function of the Second Kind
C2M2L	Component,Context,Manufacturing Model Library
CDF	Cumulative Distribution Function
CONUS	Continental United States region
CRG	Curved Regular Grid format for terrain description from OpenCRG
CSIR	Council for Scientific and Industrial Research (South Africa)
CSV	Comma-Separated Values
DCG	Definite Clause Grammar (what renders this text)
DCS	Dynamic Context Server
DEM	Digital Elevation Model for mapping
DOI	Department of the Interior
DSL	Domain Specific Language
E-M	Electro-Magnetic
EME	Electro-Magnetic Environment
EMI	Electro-Magnetic Interference
EMP	Electro-Magnetic Pulse
FFT	Fast Fourier Transform
FMI	Functional Mockup Interface

FMU Functional Mockup Unit
FP Fokker-Planck
FT Fourier Transform
GIS Geographic Information Systems
GPS Global Positioning System
GW Gravity Wave
HTML hypertext markup language (what you are reading right now!)
JPL Jet Propulsion Laboratory
JSON JavaScript Object Notation
LL log-log scale
MC Monte Carlo random number simulation
ME Maximum Entropy
MaxEnt Maximum Entropy
NASA National Aeronautics and Space Administration
O-U Ornstein-Uhlenbeck random walk process model
OSCAR Ontological System for Context Artifacts and Resources
OWL Web Ontology Language
PCC Probabilistic Certificate of Correctness
PDF Probability Density Function (for documentation see Portable Data Format)
PSD Power Spectral Density or Particle Size Distribution depending on context
RDF Resource Description Format
RF Radio Frequency
RMS Root Mean Square
SWEET Semantic Web for Earth and Environmental Terminology
SWH Significant Wave Height
SoS Superposition of Sine waves
TOPS Test Operating Procedures
TTL Turtle Triple-Store format
URL Universal Resource Locator
USGS US geological Survey
XML Extensible Markup Language
YPG Yuma Proving Grounds
YTC Yuma Test Center

References

[1] S. Bankes, D. Challou, T. Haynes, H. Holloway, J. Tierno, and C. Wentland, “META Adaptive, Reflective, Robust Workflow (ARRoW),” BAE Systems, Final Report TR-2742, 2011.
[2] P. R. Pukite, S. Bankes, and D. Challou, “stochastic_analysis.docx.”
[3] P. R. Pukite, S. Bankes, and D. Challou, “terrain_characterization.docx.”
[4] P. R. Pukite, S. Bankes, and D. Challou, “diffusive_growth.docx.”
[5] R. O’Keefe, *The craft of Prolog*. 1990.
[6] Wielemaker,J., T. Schrijvers, M. Triska, and T. Lager, “SWI-Prolog,” *Theory and Practice of Logic Programming*, vol. 12, pp. 67–96, 2012.

[7] J. Wielemaker and Z. Huang, "SWI-Prolog and the web," *Theory and Practice of Logic Programming*, vol. 8, no. 3, pp. 363–392, 2008.
[8] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann, 2011.
[9] G. Macgregor, "E-resource management and the Semantic Web: applications of RDF for e-resource discovery," *The E-Resources Management Handbook*, vol. 18, 2009.
[10] R. G. Raskin and M. J. Pan, "Knowledge representation in the semantic web for Earth and environmental terminology (SWEET)," *Computers & Geosciences*, vol. 31, no. 9, pp. 1119–1125, 2005.
[11] C. Yang, R. Raskin, M. Goodchild, and M. Gahegan, "Geospatial Cyberinfrastructure: Past, present and future," *Computers, Environment and Urban Systems*, vol. 34, no. 4, pp. 264–277, 2010.
[12] W3C, "SPARQL 1.1 Entailment Regimes." [Online]. Available: http://www.w3.org/TR/sparql11-entailment/. [Accessed: 25-Jan-2013].
[13] M. Kay, *XSLT programmer's reference*. Wrox Press Ltd., 2001.
[14] J. Wielemaker, *SWI Prolog Reference Manual 6.2. 2*. BoD–Books on Demand, 2012.
[15] PHP Group, "PHP: hypertext preprocessor," 2001.
[16] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, "Design patterns: Elements of reusable object-oriented software," *Reading: Addison-Wesley*, 1995.
[17] D. Mumford and A. Desolneux, *Pattern Theory: The Stochastic Analysis Of Real-World Signals*. A K Peters, Ltd., 2010.
[18] ATEC, "US Army Test and Evaluation Command Test Operations Procedure 1-1-011," 1981.
[19] Automotive Directorate, "Test Operations Procedure (TOP) 1-1-010 Vehicle Test Course Severity (Surface Roughness)," ATEC, Aberdeen Proving Ground, MD, Final TOP 1-1-010, 2006.
[20] Automotive Directorate, "Test Operations Procedure (TOP) 01-1-011A Vehicle Test Facilities at Aberdeen Test Center and Yuma Test Center," US Army Aberdeen Test Center, Aberdeen Proving Ground, MD, Final TOP 01-1-011A, 2012.
[21] podaac@podaac.jpl.nasa.gov <podaac@podaac.jpl.nasa.gov, "PO.DAAC." [Online]. Available: http://podaac.jpl.nasa.gov/. [Accessed: 09-Apr-2012].
[22] C. of E. US Army, "Wave Information Studies." [Online]. Available: http://wis.usace.army.mil/hindcasts.shtml. [Accessed: 31-May-2012].
[23] T. E. Vanhecke, "Zotero," *Journal of the Medical Library Association: JMLA*, vol. 96, no. 3, p. 275, 2008.
[24] Jonathan Cameron, Abhi Jain, Terry Huntsberger, Garett Sohl, and Rudranarayan Mukherjee, "Vehicle-Terrain Interaction Modeling and Validation for Planetary Rovers," NASA JPL, Dartslab, JPL Publication 10-15, 2009.
[25] GitHub, "OpenRefine (OpenRefine)." [Online]. Available: https://github.com/OpenRefine. [Accessed: 25-Jan-2013].
[26] "GRefine RDF Extension." [Online]. Available: http://refine.deri.ie/. [Accessed: 25-Jan-2013].
[27] Center for History and New Media, "Zotero Quick Start Guide." [Online]. Available: http://zotero.org/support/quick_start_guide.
[28] J. Ritterbush, "Supporting Library Research with LibX and Zotero," *Journal of Web Librarianship*, vol. 1, no. 3, pp. 111–122, 2007.
[29] T. Huang, S. Hardman, A. Bingham, A. Takagi, Q. Chau, and M. Gangl, "Building the Next Generation Data Management and Archive System," presented at the AGU Fall Meeting Abstracts, 2009, vol. 1, p. 1061.
[30] Atlanta Engineering, "Calculate Vehicle Speed from Skid Mark Distance." [Online]. Available: http://www.atlantaeng.com/aes_calculator_vehicle_speed_combined.html. [Accessed: 08-Nov-2012].
[31] Army, "Research, Development, Test and Evaluation of Materiel for Extreme Climatic Conditions," APD, Headquarters, Dept of the Army, Army Regulation 70-38, Sep. 1979.
[32] ISO, "ISO 12944-5-2007." [Online]. Available: http://www.scribd.com/doc/56997496/ISO-12944-5-2007. [Accessed: 02-Nov-2012].
[33] O. LivingSteel, "Corrosion - Living Steel - Living Steel." [Online]. Available: http://www.livingsteel.org/corrosion-3. [Accessed: 02-Nov-2012].
[34] Wikipedia, "Sea state - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Sea_state. [Accessed: 21-Jun-2012].
[35] S. Machlup, "Noise in Semiconductors: Spectrum of a Two-Parameter Random Signal," *Journal of Applied Physics*, vol. 25, no. 3, pp. 341–343, 1954.

[36] Wikipedia, “Ornstein–Uhlenbeck process - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Ornstein%E2%80%93Uhlenbeck_process. [Accessed: 19-Oct-2012].
[37] C. Becker and P. S. Els, “GEROTEK data from CSIR for C2M2L program.” .
[38] G. Gonçalves and J. Santos, “Propagation of Dem Uncertainty: An Interval Arithmetic Approach.,” in *XXII International Cartographic Conference, Spain*, 2005.
[39] USGS, “USGS/EROS Find Data/Products and Data Available/DEMs.” [Online]. Available: http://eros.usgs.gov/#/Find_Data/Products_and_Data_Available/DEMs. [Accessed: 24-Apr-2012].
[40] VTP, “DEM (digital elevation model).” [Online]. Available: http://vterrain.org/Elevation/dem.html. [Accessed: 12-Nov-2012].
[41] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “Dbpedia: A nucleus for a web of open data,” *The Semantic Web*, pp. 722–735, 2007.
[42] P. Yue, J. Gong, and L. Di, “Augmenting geospatial data provenance through metadata tracking in geospatial service chaining,” *Computers & Geosciences*, vol. 36, no. 3, pp. 270–281, 2010.
[43] M. C. Bowers, “The distributions of seasonal river flows: lognormal or power-law?,” MSc, Purdue.