

# Knowledge Based Environmental Modeling

**Abstract:** This paper describes a modeling architecture based on logical building-blocks well suited for comprehensive environmental modeling.

## 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 details of the vehicle design. A vehicle has to endure and even thrive in extreme conditions to earn the title of ruggedized.

The approach that we outline below extends from research describing the elements necessary to create workflow architectures for vehicle design<sup>1</sup>. In particular, the building blocks that originated from design-centric semantic web and knowledge-based reasoning were deemed general enough to find applicability to environmental modeling.

Based on the general approach to design, we applied the same knowledge-based facets to environmental modeling (see Table 1 below). Starting from the modeling level, we consider abstractions, organization, search, and reasoning to improve the convenience and usability across the suite of models.

As models are quantitative and often statistical, 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.

Table 1 : Knowledge-based modeling facets

	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 views of environmental contexts
Organization	 Semantic web ontologies such as SWEET	Classification, maintenance, 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

The supplemental view is a dynamic context server (DCS) which provides an environment for handling interactive applications such as guided workflows, artifact viewing, and reusable web services. OSCAR's capabilities will not overlap with the dynamic server's capabilities and in

fact the two complement each other. The two views can be integrated through a conventional web-service front-end.

The DCS has been prototyped with a breadth-first view first to make sure that we have all the semantic, ontological, logical, and mathematical capabilities in place.

### General Architecture of a Dynamic Context Server

- Integration with triple-store semantic data
  - RDF and simple formats such as Turtle used as input files for data and knowledge
  - SWEET provides ontological classification where possible
- Knowledge-based logic formulation
  - Rules interact with the triple-store description logic
  - Entailment – creating rules that abstractly appear as triple store
- Semantic Web servicing
  - Queries as model services
  - Queries to generate algorithms and code
  - Queries to generate artifacts and metrics
- Workflows
  - Query-based search and navigation
  - Guided workflow starting from requirements

### Architecture Features

- Pattern architecture
  - Many context models have similar representation or archetypes
    - e.g. PDF patterns use functional mapping on lists
    - Similar format allows reusable artifacts, such as graphs
  - Models calibrated via parameters for different locales
  - Declarative programming facilitates pattern-matching
- Graphing of artifacts
  - From models or data
    - PDF and CDF
    - Power Spectral Density
    - Expected value plots
      - Annual, diurnal
      - Growth curves
  - Linear or log scales
  - Noise filtering
  - Inspection of data points
  - Panning and scaling
- Domain-specific language for list processing
  - List of X-Y pairs
  - List of X-Y1,Y2,... n-tuples
  - Math operations on lists
    - Constructors
      - Linear range
      - Log range
  - Dot product
  - Mapping on lists
    - Functions
    - Scaling
  - Convolution

- Pair Correlation / Auto-correlation
  - Z-difference
  - Integral / Derivative
  - Histogram
  - Direct translation of tuple-lists to graphs
- Domain-specific language for complex-number processing
  - Arithmetic
    - Infix operators for multiply, addition, division, power, etc
    - Used for concise semi-Markov terrain profile modeling
  - Built-in FFT for PSD calculation
- Integration with general statistical functions
  - R statistics package
    - Uses similar format for mapping against data lists
    - i.e. Statistical operators act on the entire list
  - Provides access to less common functions such as Bessel
- Random number generation
  - PDF generators for
    - Uniform
    - Exponential
    - Bessel
    - Rayleigh
    - Fat-tail
  - Profile generators
    - Random walk based
      - Markov
      - Semi-markov
    - Superposition of sines
- Reference and citation knowledge capture
  - Integration with Zotero citation management system
    - SWEET keyword categorization
    - Generation of RDF stores
- Rules for searching and presenting knowledge
  - Based on ontological terms
  - General content search
  - Custom input and output data processing
    - XML and HTML
    - JSON
    - CSV
- Diagramming
  - AT&T Graphviz compatibility with SVG
  - Closure of rules as a directed graph.
  - Hierarchy and cloud diagrams
- Generators for code production and data
  - Tabulation of data, i.e. connectors for spreadsheet
  - Symbolic representations of algorithms
    - Used for on-line calculation
    - Translated to C-code, Python, etc
- GUI development
  - Uses definite clause grammars (DCG) to generate valid HTML
  - Declarative style of programming, similar but much more concise and powerful than XSLT
- Portability
  - Windows
  - Linux
  - Cloud environment
- Maintenance of library

- Browsing
- Site roadmap
- Documentation
- Statistical usage
- Password control for administration of repository

#### Physical Domains

- List of physical domains
  - Land
  - Atmosphere
  - Aquatic
- Deterministic and stochastic representations
  - Stochastic terrains characterized by PSD's
  - Deterministic obstacles

#### Installation

##### Source Code

All source code is available on SVN under the **context** branch

browser: [https://babelfish.arc.nasa.gov/trac/avm\\_performers/browser/context](https://babelfish.arc.nasa.gov/trac/avm_performers/browser/context)

checkout: [https://babelfish.arc.nasa.gov/svn/avm\\_performers/](https://babelfish.arc.nasa.gov/svn/avm_performers/)

#### Packages Required

1. SWI Prolog for Linux, Windows, or Mac <http://www.swi-prolog.org/Download.html>
2. R Statistics package <http://www.r-project.org/>
3. AT&T Graphviz graphics package <http://www.graphviz.org/>
4. Firefox preferred, but Google Chrome adequate as a browser. Internet Explorer won't work.
5. (possible) Geospatial indexing library <http://www.swi-prolog.org/pldoc/package/space.html>

Make sure to install all the add-on packages for SWI. The prolog runtime will need to know the path to R and Graphviz. The geospatial indexing package needs to be added to the SWI Prolog build to work.

Once the SVN is checked-out, go to **context\Ontology\reasoners\prolog\proto\_context\_server** directory to start the Dynamic Context Server.

- **run.pl** (Windows registered)
- **run.sh** (Linux command line)
- **nohup-run-cloud** (running on a Cloud server as a job)

The port is 3020 for local clients and 80 for cloud configured. If you have the DCS set up on the machine you are using to access this Wiki then click on the following link:

<http://localhost:3020>

August 7, 2012

An authorization login is required to load the ontological context data (menu item **Repository/Load Context Data**). This can be reset by removing the users.db file in the store directory.

Cloud computing. Server.

## T

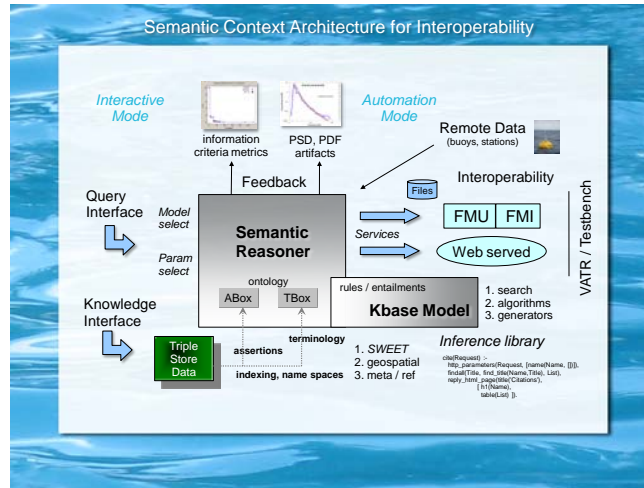
*Table 2: The set of Domain Specific Language elements for environmental modeling covers math and logic functionality*

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

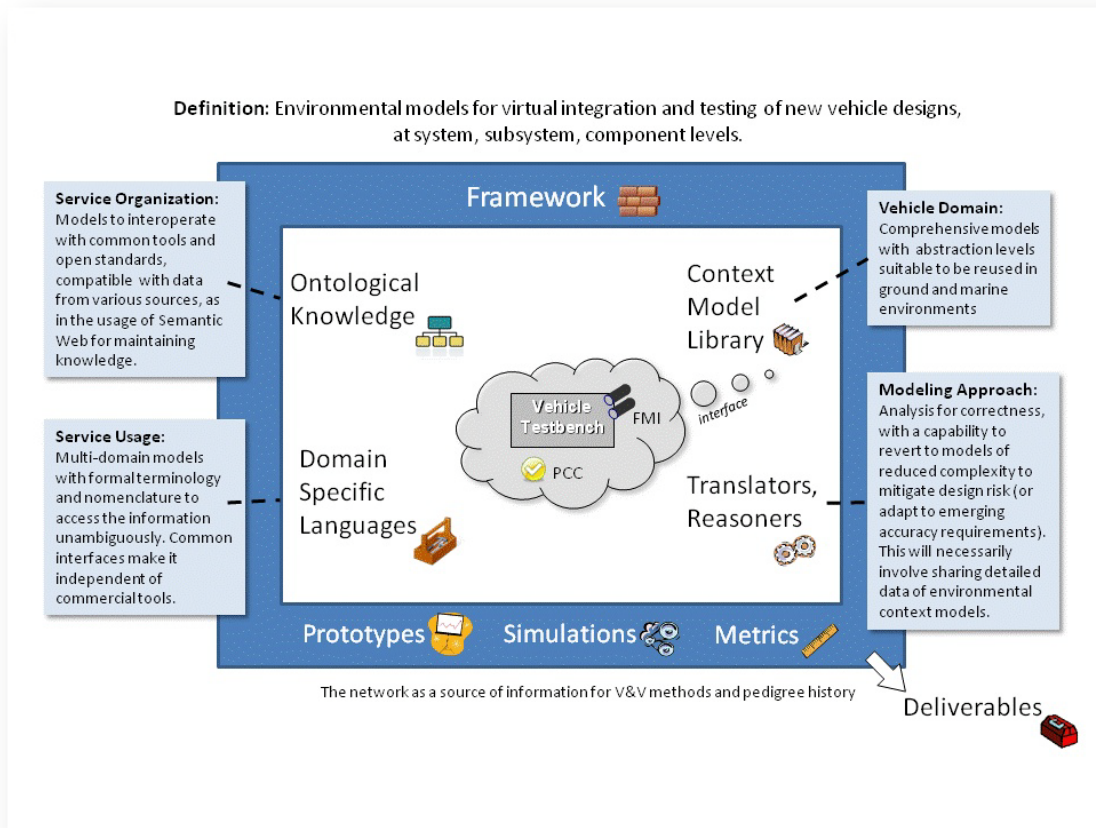
a  
a  
a  
a  
a

The geospatial functionality is very useful for inferring 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.

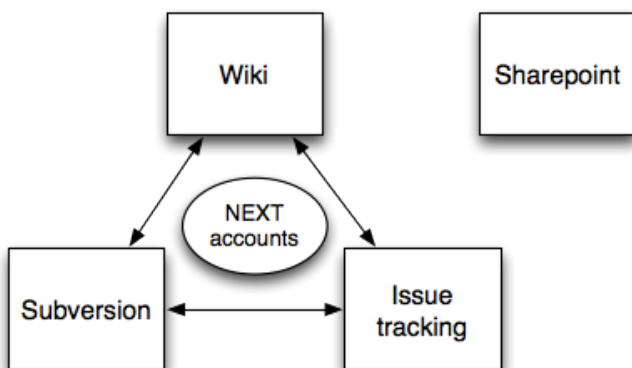
## Knowledge Based Representations



## Summary



AA



August 7, 2012

## References

---

<sup>1</sup> S. Banks et al., *META Adaptive, Reflective, Robust Workflow (ARROW)*, Final Report, Phase 1B (BAE Systems, 2011), [http://www.darpa.mil/Our\\_Work/TTO/Programs/AVM/AVM\\_Meta\\_Final\\_Deliverables.aspx](http://www.darpa.mil/Our_Work/TTO/Programs/AVM/AVM_Meta_Final_Deliverables.aspx).