Ontological System for Context Artifacts and Resources (OSCAR) [1]

Revision A

Dr. Rudranarayan Mukherjee
Thomas Huang
Nga Quach

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91109-8099
California Institute of Technology

TABLE OF CONTENTS

1. INTRODUCTION

JPL is a member of a larger BAE led team that has successfully developed and won a proposal in response to the DARPA-BAA-11-47 Component, Context, and Manufacturing Model Library 1 (C2M2L-1) form by the Tactical Technology Office, DARPA. The following text taken from BAA is provided to the reader for understanding the sponsor's needs:

> *The DARPA META program is developing a formal metalanguage for the representation of complex cyber-electro-mechanical systems, a set of design tools and metrics for performing trade-space exploration, and a set of verification tools for stochastic formal verification of large system designs…*
>
> *One of the principal thrusts of this program is on the development of overall environment models (e.g., terrain, atmosphere, water, etc.) affecting drivetrain and mobility subsystems to include amphibious considerations…*
>
> *Terrain models are expected to represent the surface/fluid that an amphibious infantry fighting vehicle would traverse, ranging from paved road surfaces to rocky, mountainous terrain, slope, discrete obstacles (such as step climbs, v-ditches, etc.), mud, sand, snow, and water fording (both salt water in a ship-to-shore deployment typical of a Marine Air- Ground Task Force, as well as fresh or coastal water for river/lake/etc. fording)…*
>
> *Particulates models of interest include atmospherically-borne particulate matter such as dust, sand, snow, ice particles, water-borne particulates (when submerged during amphibious transit), and volcanic ash, and their interaction with the mobility and drivetrain subsystems….*
>
> *The software tools, documentation, specifications, and sample models being produced under the META, iFAB, and vehicleforge.mil efforts are being developed as open-source software.*

2. JPL RESPONSIBILITIES

As described in the proposed TERRAIN ONTOLOGY AND GRANULAR MEDIA MODELING FOR MOBILITY CONTEXT task plan, the ontology portion of the task requires the following:

- JPL will adapt its SWEET ontologies to provide a formal ontology to formulate, design, and implement architecture for search/locating, storing, retrieving and maintaining an ontology of the context models.

- JPL will work with the BAE C2M2L-1 team to formulate, design, and implement architecture to search, locate, store, retrieve and maintain a semantic-based digital library for the context models. The context library will store all of the contexts developed over the life of this project. The library will be based on an expansion of the SWEET ontologies.

The final deliverable is a knowledgebase library system for context models. An archival system that is driven by the backend ontology designed for environmental context along with concepts for common archival transactions.

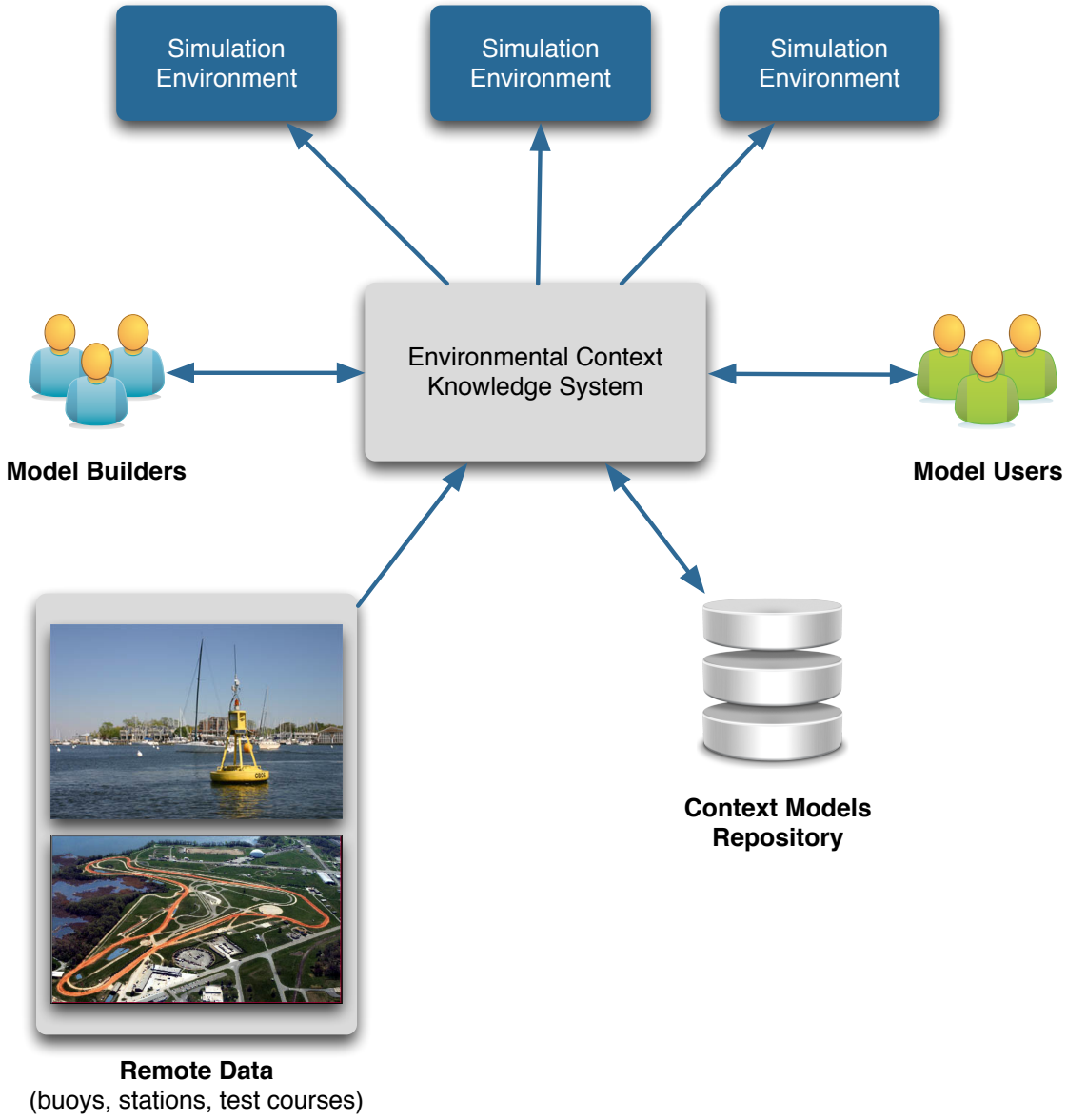


Figure 1 High-Level System Architecture

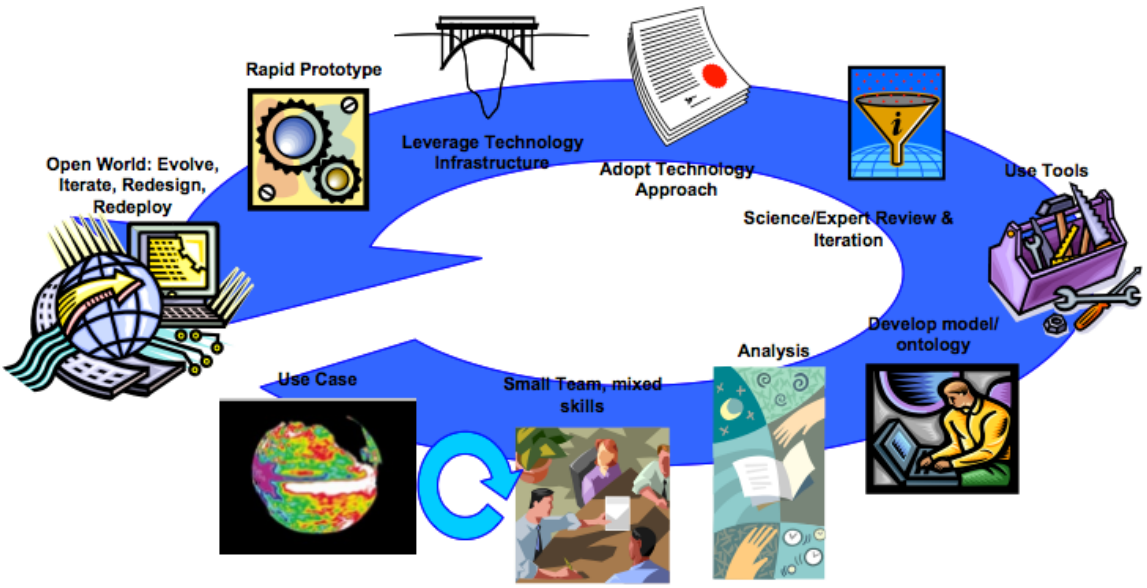
3. METHODOLOGY AND DEVELOPMENT PROCESS

Methodology is a guideline system for solving problem. A process is a series of actions or steps taken to achieve an end. Both of these are essential in development of any data system. This section describes the methodology, the process, and some of the considerations were made in developing this knowledgebase system.

3.1 Semantic Web Development Methodology

Developing an ontological system requires careful planning and promotes teaming. An iterative methodology was adopted from the beginning with the following steps

- Use Case development
- Small Team with maxed skills
- Analysis
- Develop model / ontology
- Use Tools
- Science/Expert Review & Iteration
- Adopt Technology Approach
- Leverage Technology Infrastructure
- Rapid Prototype
- Open World: Evolve, Iterate, Redesign, Redeploy



Capturing Complex Relationship and Taxonomy

Through working closing with the sponsor and building a team with mixed skill sets: project management, ontologist, system and information architect, model builder, testbed developer, and software developer, a set of use cases were rendered. These use cases enables discussion on taxonomy and relationship between artifacts.

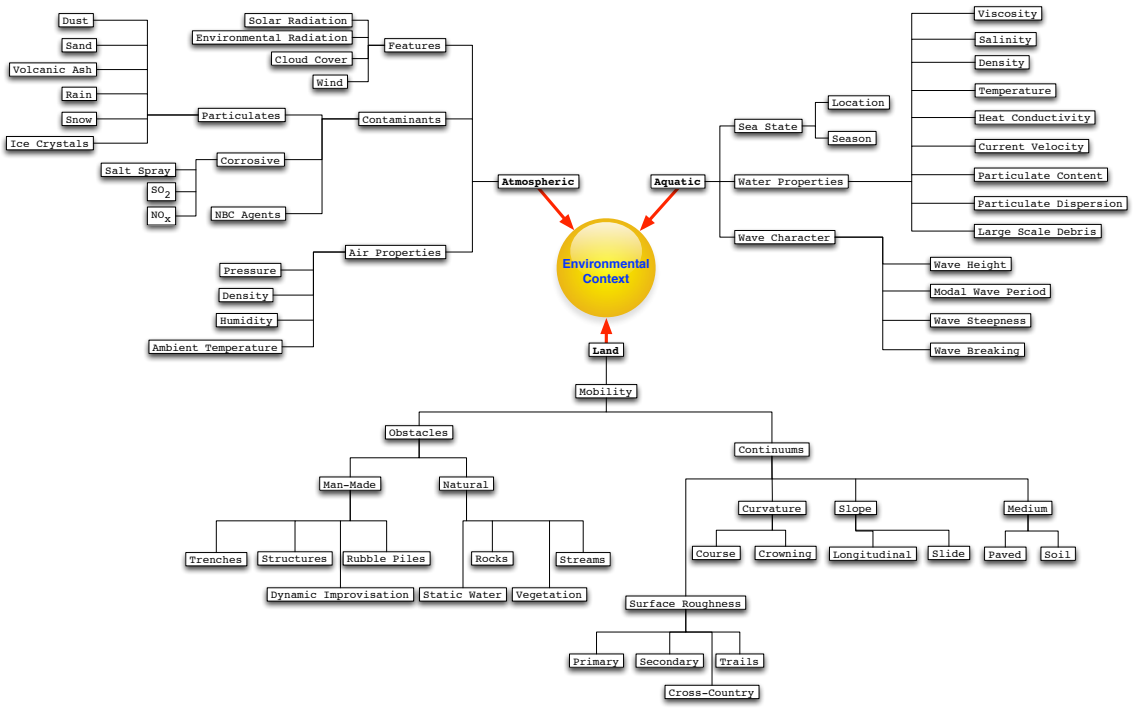


Figure 2 Environmental Taxonomy Organization

The diagram below illustrates using environmental context to link various classes of metadata together.

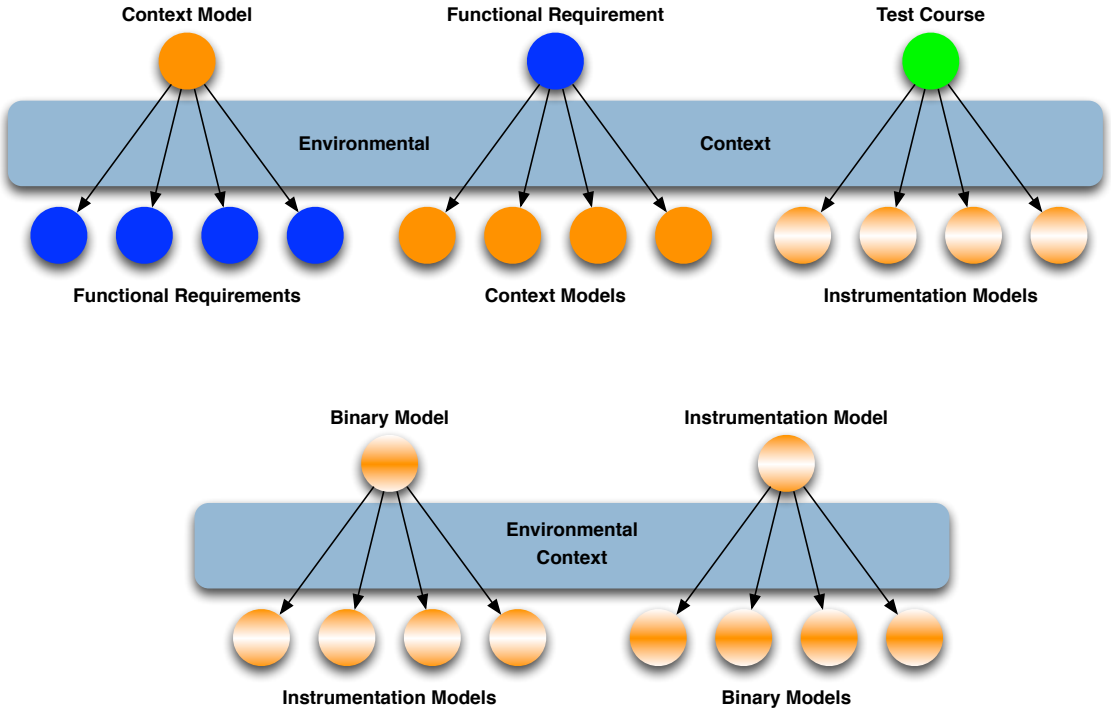


Figure 3 Linking Between Metadata and Artifacts

Leverage SWEET Ontologies

The Semantic Web for Earth and Environmental Terminology (SWEET) ontologies (http://sweet.jpl.nasa.gov) is a widely adapted by the Earth Science community as the de facto ontology for modeling earth environment. The ontologies were initially developed to capture the relationships between keywords defined by the Global Change Master Directory (GCMD) (http://gcmd.jpl.nasa.gov). The SWEET ontologies enables scalable classification of Earth system concepts and has expanded for support space science in recent years. The current release, SWEET 2.3, is highly modular with over 6000 concepts and 200 separate ontologies.

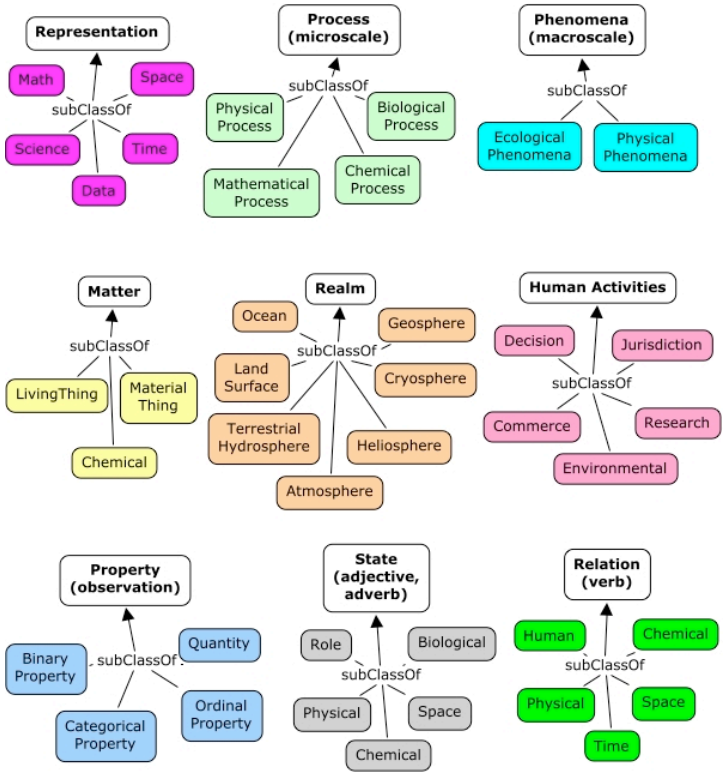


Figure 4 SWEET Ontologies

SWEET ontologies is designed for model natural environment. With terrain modeling also include manmade features and obstacles; the first major milestone of this task is in development of the backend ontology that works with the FANG requirements and captures the expectations of the model and testbed developers. SWEET, as the upper level ontology, promotes the following steps in adapting the richness of SWEET:

- Import – import only the ontologies required
- Expand – introduce new concepts and attributes specific to the application domain

- Specialize – extend and specialize SWEET concepts according to the application domain.

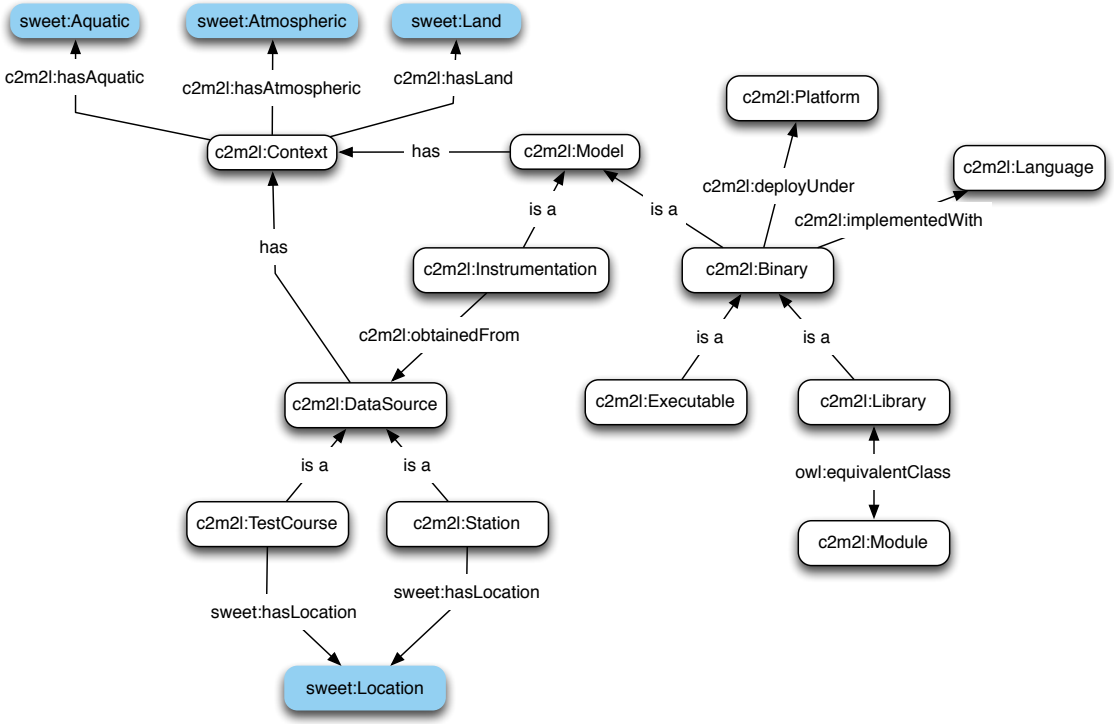


Figure 5 Leverage the SWEET Ontologies

Tools For Ontology Development

A collection of industry-standard designing tools was used to facilitate the design of the OSCAR ontology. These tools are used for documentation and validation of the ontology in development. Below is a list of the tools used

- Protégé
- TopBraid Composer
- SemanticWorks
- OxygenXML

4. SYSTEM ARCHITECTURE

The Ontological System for Context Artifacts and Resources (OSCAR) is designed from the ground up in accordance to the RESTful service architecture. It promotes abstraction and separation of concerns, by considering objects within the system as resources and each object has a set of operations. The key components of OSCAR includes

- Framework for handling RDF/OWL data including reasoning
- Repository for reliable store and retrieval of data
- Fast indexed search
- Security architecture to support authentication, authorization, and communication

Java™ is used for the implementation of OSCAR, which maximize OSCAR's portability. Over the years, Free and Open Source Software (FOSS) has proven itself to be the winning card for developing any software system. Examples of successful FOSS projects include Linux, Apache Software Foundation, Eclipse Foundation, Android Open Source Project, Perl, PHP, Python, etc. OSCAR is designed and developed to leverage from industry-standard open source components, which frees our sponsor from any long term maintaining licensing and vendor lock-in

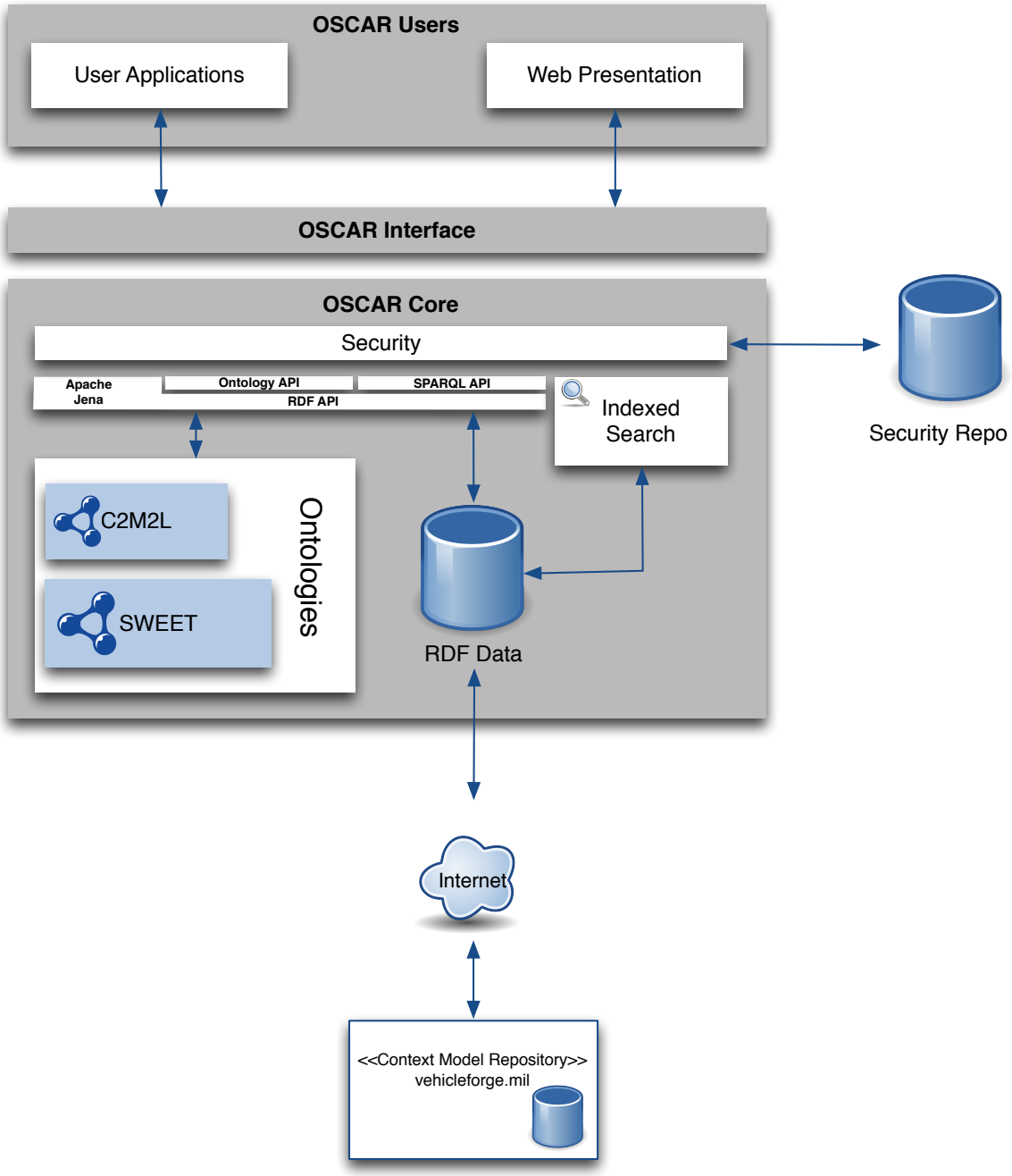


Figure 6 OSCAR Webservice Architecture

OSCAR is designed with the following goals

- A semantic web based system
- Leverages successful JPL data service projects
- RESTful service architecture
- Ontology based on SWEE and adapted to specific mobility contexts for C2M2L

- Archival and distribution - Create, Read, Update, and Delete (CRUD) context model artifacts
- Indexed and guided searches
- Auto linking between artifacts and resources
- Pluggable backend triplestore
- Pluggable security model

The OSCAR Software Stack

- Groovy – The dynamic scripting language built for the Java™ Virtual Machine (JVM)
- Grails – The dynamic web framework built using the Groovy scripting language
- Apache Jena – The RDF/OWL framework for handling RDF data
- Apache Fuseki – The opensource triplestore
- Apache Solr – The indexed search engine
- LDAP (Optional) – For user authentication and authorization

5. SYSTEM CAPABILITIES

In this production release of OSCAR, the system delivers the following capabilities captured in the screenshots below

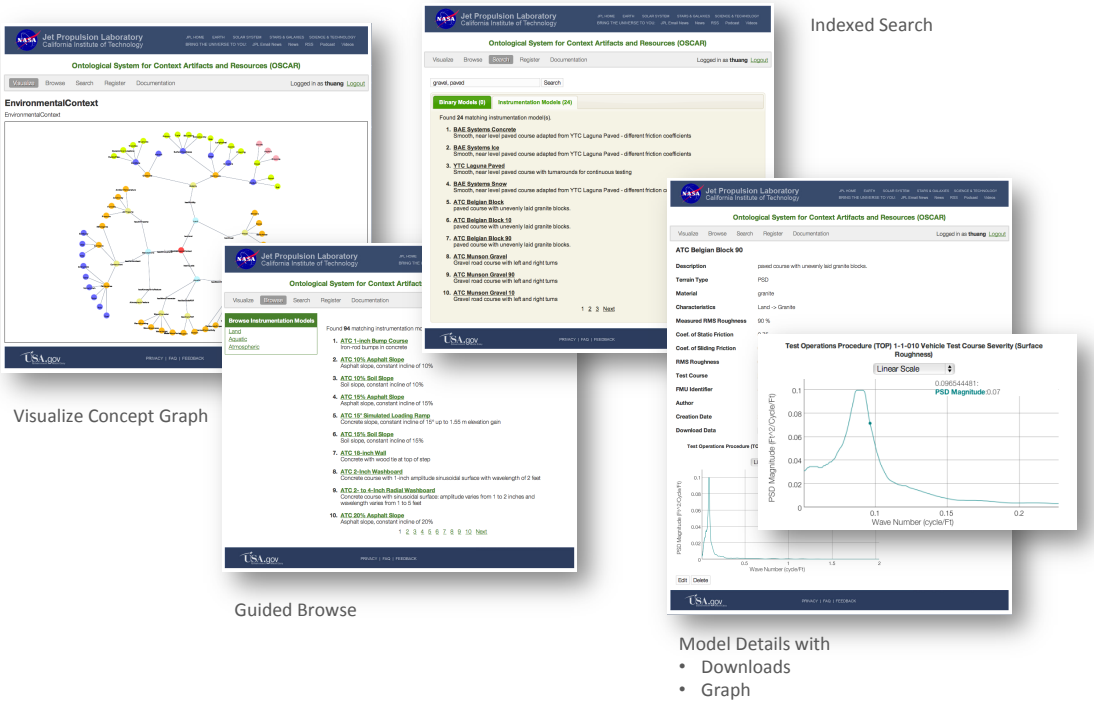


Figure 7 OSCAR Screenshots 1

1. Visualize concept graph. This graph illustrates who information is mapped within OSCAR. The users can zoom in/out of the visual, click on a concept for closer inspection, and freely reposition the visual graph.

2. Guided Browse. All models and requirements are paginated and users are guided by working with the left panel filter

3. Index Search. This offers a horizontal search across all context and artifact descriptions. This enables users to quickly lookup models according to specific context or key phases.

4. Model Detail. Each model has a detail description page. Models are divided into instrumentation and binary models. Instrumentation models are tabular data obtained from data center or produced by model builder. Binary models are either library or executable. They are platform-specific with inputs and outputs. For PSD instrumentation data, the model detail page also provides visual plot of the data.

The user can download the model from this page. For tabular data, OSCAR provides JSON, CSV, and tab-delimited model data. For binary models, OSCAR delivers the model in a compressed TAR file.

Below consists of additional screenshots of OSCAR

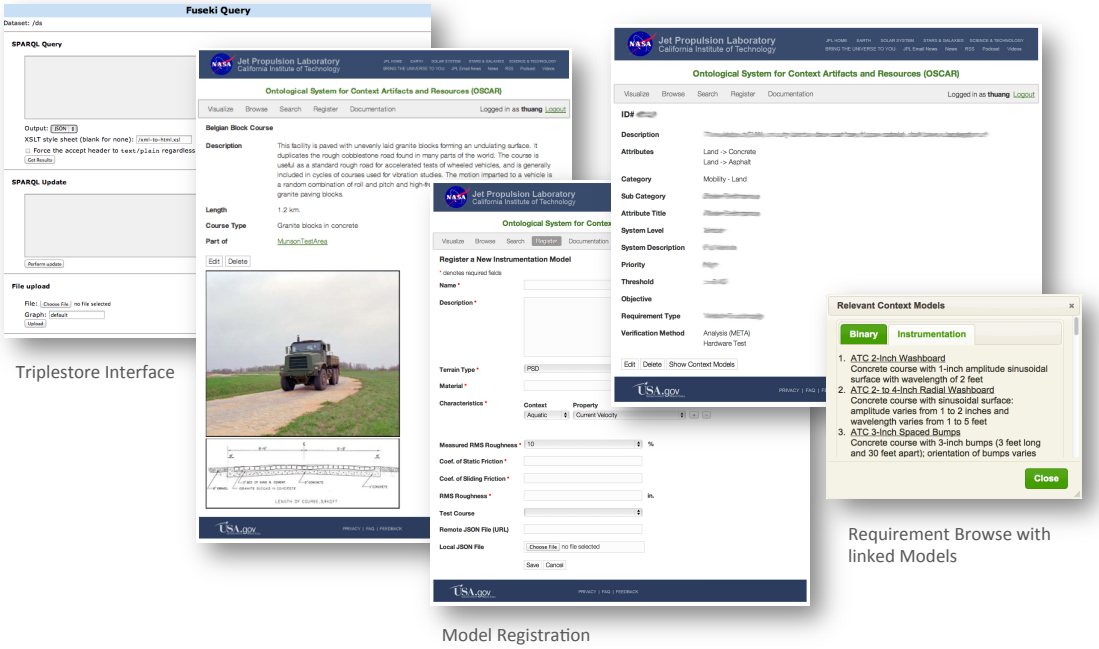


Figure 8 OSCAR Screenshots 2

1. Triplestore interface. This is an interface to the Apache Fuseki triplestore. This web interface enables the user/administrator to interact directly with the triplestore via SPARQL, load and export data into the store.
2. Test Course. Each instrumentation model obtained from a test course should have an associated test course data sheet, which provides detailed geographic information on the test course.
3. Model Registration. This is a form-based interface for model builder to register models. The OSCAR system minds the context and description of the model to enable it to automatically link to test courses and requirements.
4. Requirement Browse. This provides detail description on the requirement and the environmental context it associate with. Since this is a knowledgebase system, each requirement also has a list of models it links with. The enables the test bench developer to quickly identify the list of models to use to satisfy a specific requirement.

OSCAR also support range browse for models, which includes ranges like temperature and slope/grade.

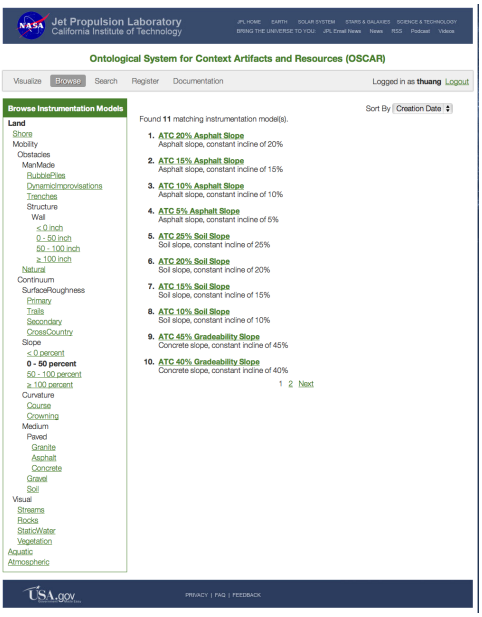


Figure 9 OSCAR Range Selection

To help user to quickly jump the to the requirement they are looking for, OSCAR user interface offers a requirement number search function.

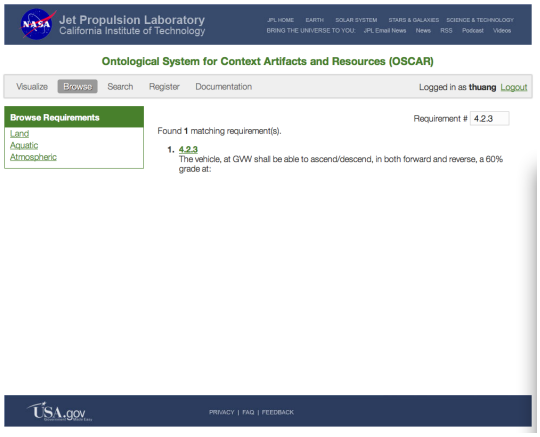


Figure 10 OSCAR Requirement Number Search

6. OSCAR INSTALLATION

6.1 Prerequisites

Prior to running OSCAR, the following must be installed and/or configured. While the entire OSCAR software stack is developed using pure Java™ technology, that is, OSCAR is very portable and can be deployed onto any platform with Java™ Runtime installed, the instructions presented here were captured on a **CentOS 5.8** platform using **T-CSH (tcsh)**, an enhanced UNIX C shell (csh). The installation is performed under the user account (**jdoe**), whose home directory is under (`/home/jdoe`)

6.2 Dependencies and Downloads

- Java 1.6.0_37
  `http://www.oracle.com/technetwork/java/javase/downloads`
- Groovy 1.8.6
  `http://dist.groovy.codehaus.org/distributions/`
- Grails 2.0.4
  `http://grails.org/Download`
- Apache Fuseki 0.2.4
  `http://archive.apache.org/dist/jena/binaries/`
- Apache Maven 3.0.4
  `http://maven.apache.org/download.html`
- Apache Solr 3.6.1
  `http://www.apache.org/dyn/closer.cgi/lucene/solr/3.6.1`

6.3 Unpack and Environment Configuration

The Java, Groovy, Grails, and Maven are the common tools/runtime environment for OSCAR. It is recommended to have them installed under a global location, such as `/usr/local/`. For this installation guide, the three packages will be installed under a local directory (`/home/jdoe/local`)

```
% pwd
/home/jdoe
% mkdir local
% cd local
```

6.4 Install Java

```
% pwd
/home/jdoe/local
% chmod ugo+x $HOME/Downloads/jdk-6u37-linux-i586.bin
% $HOME/Downloads/ jdk-6u37-linux-i586.bin
% setenv JAVA_HOME $cwd/ jdk1.6.0_37/
% setenv PATH ${JAVA_HOME}/bin:${PATH}
```

6.5 Install Groovy

```
% pwd
/home/jdoe/local
% unzip $HOME/Downloads/groovy-binary-1.8.6.zip
% setenv GROOVY_HOME $cwd/ groovy-1.8.6
% setenv PATH  ${GROOVY_HOME}/bin:${PATH}
```

6.6 Install Grails

```
% pwd
/home/jdoe/local
% setenv GRAILS_HOME $cwd/grails-2.0.4
% setenv PATH ${GRAILS_HOME}/bin:${PATH}
```

6.7 Install Apache Maven

```
% pwd
/home/jdoe/local
% tar -zxvf $HOME/Downloads/apache-maven-3.0.4-bin.tar.gz
% setenv MAVEN_HOME $cwd/apache-maven-3.0.4
% setenv PATH ${MAVEN_HOME}/bin:${PATH}
```

6.8 Up and Running

This Alpha release of OSCAR is delivered as semi-packaged distribution to simplify the installation process. A detailed installation procedure will be delivered as part of the OSCAR's final delivery.

This distribution is delivered as a self-contained compressed tarball (`jpl-oscar-1.0.tar.gz`). This installation is performed under `/home/jdoe/dev` directory

```
% pwd
/home/jdoe
% mkdir dev
```

```
% cd dev
% tar -zxvf jpl-oscar-1.0.tar.gz
% ls
C2M2L-1
% cd C2M2L-1
% ls
apache-solr  apache-solr-3.6.1     jena-fuseki-0.2.4
ontologies  oscar-web
```

Running Apache Fuseki

```
% cd jena-fuseki-0.2.4
% mkdir tdb
% sh fuseki-server --update --loc=tdb /ds
```

To check Apache Fuseki is running, point your browser to
`http://localhost:3030`

Running Apache Solr

```
% pwd
/home/jdoe/dev/C2M2L-1
% cp -r apache-solr/oscar apache-solr-3.6.1/example/
```

- Edit file `apache-solr-3.6.1/example/oscar/conf/data-config.xml`
- Locate the '`baseDir`' attribute and change its value to `<oscar-web>/triples`, where `<oscar-web>` is the fully qualified path to the location of the oscar-web directory.
  e.g.
  `baseDir="/home/jdoe/dev/C2M2L-1/oscar-web/triples"`

```
% pwd
/home/jdoe/dev/C2M2L-1
% cd apache-solr-3.6.1/example/
% java -Dsolr.solr.home=oscar -jar start.jar
```

To check Apache Solr is running, point your browser to
`http://localhost:8983/solr/`

Running OSCAR web service

```
% pwd
/home/jdoe/dev/C2M2L-1
% cp oscar-rdf/* oscar-web/triples
```

- Edit bootstrap file: `oscar-web/grails-app/conf/BootStrap.groovy`.
- Locate line 37, which should starts with '`rdf: [`'
- Update the next 9 lines to reflect the location of the ontologies and the RDF data files. This distribution is preconfigured to assume the root location is `/home/jdoe/dev/C2M2L-1/`

```
e.g.
rdf: [
   "file:///home/jdoe/dev/C2M2L-
1/ontologies/sweet_2.3/sweetAll.owl",
   "file:///home/jdoe/dev/C2M2L-1/ontologies/ContextModel.rdf",
   "file:///home/jdoe/dev/C2M2L-1/ontologies/Course.rdf",
   "file:///home/jdoe/dev/C2M2L-
1/ontologies/EnvironmentalContext.rdf",
   "file:///home/jdoe/dev/C2M2L-1/ontologies/Requirement.rdf",
   "file:///home/jdoe/dev/C2M2L-1/oscar-
web/src/resources/triples/test_courses.rdf",
   "file:///home/jdoe/dev/C2M2L-1/oscar-
web/src/resources/triples/binary_models.rdf",
   "file:///home/jdoe/dev/C2M2L-1/oscar-
web/src/resources/triples/instrumentation_models.rdf",
   "file:///home/jdoe/dev/C2M2L-1/oscar-
web/src/resources/triples/requirements.rdf"
]
```

```
% pwd
/home/jdoe/dev/C2M2L-1
% cd oscar-web
```

Download all dependencies using Apache Maven and install the downloaded libraries for OSCAR to use.

```
% mvn dependency:copy-dependencies
% cp target/dependency/*.jar lib/
```

```
Now OSCAR webservice is ready to run
% grails clean -https
% grails run-app –https
```

The server is ready with the following message

```
| Server running. Browse to http://localhost:8080/oscar or
https://localhost:8443/oscar
```

Point your browser to `https://localhost:8443/oscar`

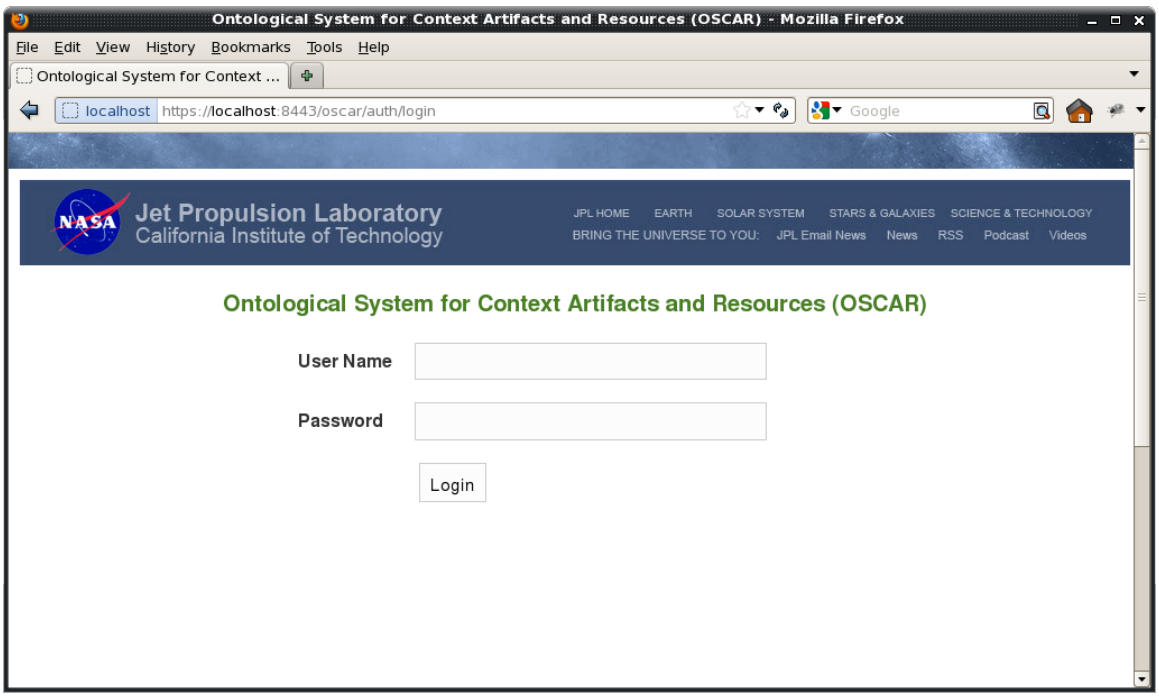


Figure 11 OSCAR landing page

Try logging in using the following account

User Name: thuang
Password: thuang

Note: to add more users, update file `oscar-web/grails-app/conf/BootStrap.groovy` starting line 23.

Firewall Configuration (Optional)

By default, CentOS 5.8 blocks all TCP points. To make OSCAR accessible within local Intranet, a firewall exception must be added for TCP port 8443. To make this change, it requires either root or sudo privilege.

```
% sudo vi /etc/sysconfig/iptables

-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 8443 -j ACCEPT

% sudo /etc/init.d/iptables restart
```

7. MANAGEMENT OF THIS DOCUMENT

Document Custodian
Rudranarayan M. Mukherjee
Mail Stop 198-235
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

Phone: 818-354-2677
Fax: 818-393-3254
Email: *Rudranarayan.M.Mukherjee@jpl.nasa.gov*