**MoSuMo: Molecular Surface Modeller**

System Documentation

Alex Gawronski

October 20, 2010

Table of Contents

[Introduction: 3](#_Toc275695670)

[Requirements: 3](#_Toc275695671)

[Installation: 3](#_Toc275695672)

[Running: 3](#_Toc275695673)

[Implementation: 4](#_Toc275695674)

[Code Style: 4](#_Toc275695675)

[Generation of Surfaces: 4](#_Toc275695676)

[Ontology Design: 4](#_Toc275695677)

[Java Servlet: 5](#_Toc275695678)

[Future Work: 6](#_Toc275695679)

# Introduction:

The Molecular Surface Modeller (MoSuMo) is SADI-based web service for generating annotated molecular surfaces. The models are derived from protein structures from the Protein Data Bank (PDB). The representations include spatial information about the shape of the protein annotated with electrostatic potential, amino acid and ligand binding site information.

# Requirements:

* Linux Operating System
* 2GB RAM (4GB+ recommended)
* Internet Connection
* Java JRE

# Installation:

1. Install Headless Linux Chimera (<http://www.cgl.ucsf.edu/chimera/download.html>)
2. Navigate into the “ChimeraPatch” folder and copy the “chimera” folder.
3. Paste the folder into the directory where Chimera was install in step 1.
4. Open MoSuMo.prop and change the “chimeraDir” to the location of the Chimera “bin” folder. ex. /home/user/chimera/bin/.
5. Install Maven or Tomcat or other Java-based web server
6. Install cURL or equivalent software

# Running:

This is an example using Maven with Jetty, cURL, and the sample input file (input.n3). Procedure will vary if different software is used.

1. Navigate to the MoSuMo directory.
2. Run “runService” to launch the service with Maven.
3. Run “curlInput” to HTTP Post the input.n3 file for the example structure “1yti”. If you wish to run a different structure, change all instances of “1yti” in input.n3 to the PDB ID of your choosing.
4. Wait for “output.rdf” to be returned.

# Implementation:

## Code Style:

The service is built from two components, the JAVA servlet and the Chimera plug-in. Chimera is a popular molecular visualization tool written in python. The JAVA servlet was built using a skeleton SADI project provided by the SADI developers.

## Generation of Surfaces:

See system documentation for the MoSuMo plug-in.

## Ontology Design:

All the ontologies described in this section were created using Protege 4.0.2. An ontology of geometric entities was developed to formalize the output of MoSuMo. This ontology contains all variations of points, lines and shapes related to each other using “has Part” and “is Part Of” rela- tionships from the Semanticscience Integrated Ontology (SIO). At the root of this ontology are “geometric entity” and “geometric entity quality”. Most of the entities in this ontology are sub classes of four sub classes of “geometric entity”: polygon, point, curve and arrowed line. Geometric entity quality holds entities that describe geometric entities, such as “straight” or “vertical”.

To describe a molecular entity surface the “polyhedral surface” entity was used. This is a three dimensional object made of faces which are part of polygons that are created from edges and points. From these entities the main component of the model is created. After the main structure was defined it had to be linked to amino acid residues, ligand binding pockets, and electrostatic potential. Most of these entities already reside in existing ontologies. Amino acid residue and molecular entity were taken from the Chemical Entities of Biological Interest (CHEBI) ontology and ligand contact from the Sequence Ontology (SO). There was no appropriate entity in any public ontology for electrostatic potential so this entity was added to the service ontology. The way that these entities were mapped to the polyhedral surface was through a “solvent excluded surface”.

With all the required entities collected, a single ontology was created to relate them to each other for use in the service. This ontology includes all the aforementioned entities as well as new entities specifically for the service related to input and output. The input class is a “Molecular Structure” which has an identifier for the PDB ID that is the only required input parameter of the service that does not have a default value.

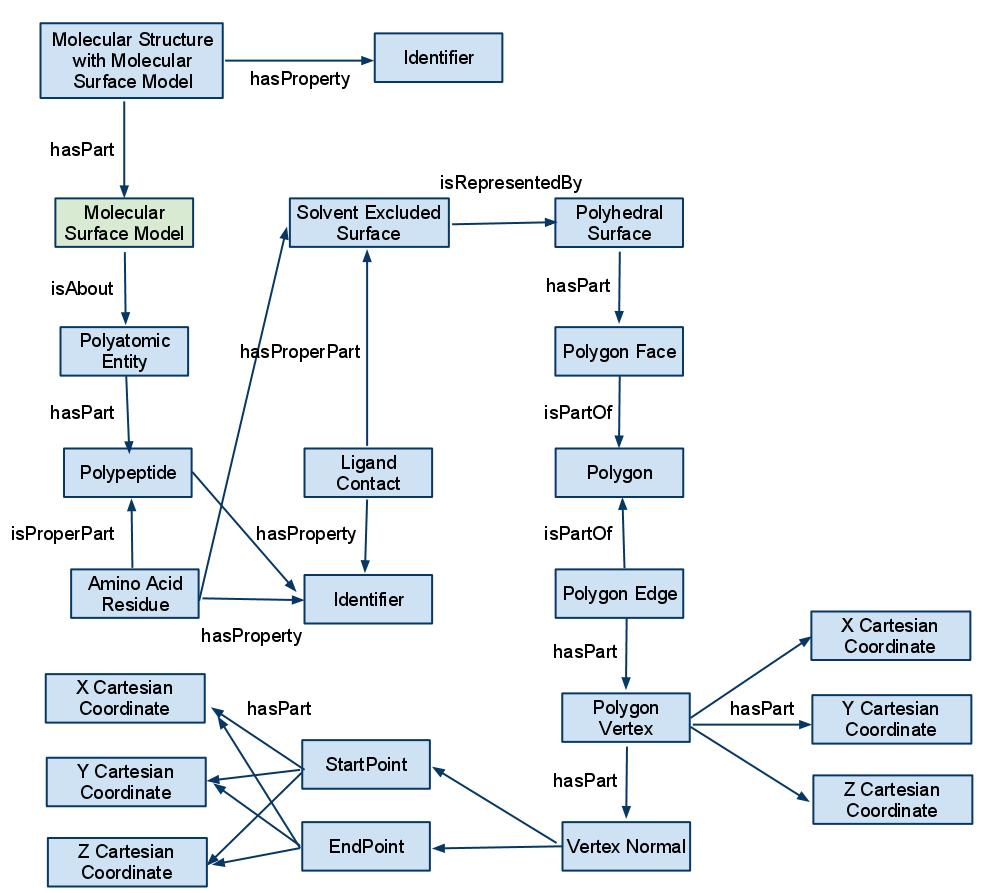


Figure : Output class of MoSuMo

The output class is a “Molecular Structure with Molecular Surface Mode” which is a sub class of “Molecular Structure”. This entity has a “Molecular Surface Model” which is defined in the ontology which “is information about” the molecular entity corresponding to the input PDB ID (Figure 1). The “molecular entity” is made up of the aforementioned biological entities and forms a solvent excluded surface that is represented by a polyhedral surface. The polyhedral surface is made up of geometric entities as defined in the geometry ontology. The two entities in the upper part of the output class figure (Figure 1) are essentially the input class, which shows how the output class is an annotated input class.

## Java Servlet:

The service servlet was created with Eclipse with the Maven 2 plug-in using Java 6. With the input and output defined the actual service had to be created. With the SADI framework this involves extending the SimpleSynchronousServiceServlet class and implementing the processInput() method. The processInput() method has two functions in the case of MoSuMo Service. The first is executing the MoSuMo program and the second is parsing its output while annotating the output class. The input parameters are obtained from the input “Resource” object. Runtime.exec() is used to produce a separate process where MoSuMo runs. The program then waits for the process to terminate and then parses the output file of MoSuMo or the error log if the output file does not exist.

The output file contains a tab delimited list of all the vertices followed by all the faces. The vertices have amino acid residue data, followed by spatial coordinates, then electrostatic potential values and finally ligand pockets if applicable. The faces are simply indices to the vertices in the first part of the file. When the file is parsed the data is transformed into “Resource” objects which are then related to each other using “Property” objects. The algorithm creates appropriate “Resource” objects (residues, vertices, etc.) in the order they come and maintains them in several hash maps. This insures there are no duplicates as well as allowing the faces to be quickly mapped to vertices when the parser reaches them. The molecular surface model is the sum of all the resources and properties and this model is then related to the output class, which is also a resource, through a “has part” property. Once this is completed the SADI framework creates the RDF and returns it to the user through the web server.

# Future Work:

* Make DelPhi parameters easier to modify by the user
* Resolve issues with model size being too large