Develops and maintains open source software applications

- Toxtree 2.1.0 – toxic hazard estimation, 12 modules
- Toxmatch 1.06 – A chemical similarity evaluation tool
- Online QMRF repository http://qsardb.jrc.it
- Ambit, Ambit XT
- Partner in OpenTox FP7 project
- Partner in CADASTER FP7 project
• **Objective:** to develop a framework that provides an unified access to:
  – toxicity data,
  – predictive models,
  – procedures supporting validation and additional information that helps with the interpretation of predicted results.

• European Commision Framework Programme 7, HEALTH-2007-1.3.3

• 11 partners
Why integration framework for predictive toxicology?

• What we would like to do:
  – Build, use, validate and compare multiple models
  – Reliable reproduce models from the literature
  – Merge data from different sources (files, databases)
  – Find all models available for certain endpoint
  – More ...
Why integration framework for predictive toxicology?

- **Challenges:**
  - Chemical structures
    - Might be ambiguous
    - Might be error prone or time consuming to reproduce from publications
  - Data
    - **Multiple formats,**
    - **Implicit semantics,** often buried in human readable documentation only
  - Models
    - **Tens of thousands** available, in software or in publications
    - Multiple software solutions, mostly incompatible
    - Predictions **reproducibility** is time consuming and often hard to achieve
    - Automatic **comparison of prediction results** difficult
## Framework design rationales

<table>
<thead>
<tr>
<th>User Requirements</th>
<th>Software Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Umambiguous data</td>
<td>⇒ <em>formal way of representing information about data</em></td>
</tr>
<tr>
<td>Unambiguous access</td>
<td>⇒ <em>well-defined interfaces</em></td>
</tr>
<tr>
<td>Transparency of computational tools</td>
<td>⇒ <em>formal way of representing information about methods, well-defined interfaces</em></td>
</tr>
<tr>
<td>Variety of user groups</td>
<td>⇒ <em>simplicity and modularity of design</em></td>
</tr>
<tr>
<td>Need to integrate various resources (e.g., databases, prediction methods, models, ...) to make meaningful predictions</td>
<td>⇒ <em>distributed architecture, interoperability</em></td>
</tr>
<tr>
<td>Need to integrate biological information</td>
<td>⇒ <em>again, modularity of design, extensibility</em></td>
</tr>
</tbody>
</table>
The framework

- OpenTox API
  - The way applications talk to each other
  - The way developers talk to applications
  - [http://opentox.org/dev/apis/api-1.1](http://opentox.org/dev/apis/api-1.1)
- The basic building blocks:
  - data, chemical structures, algorithms and models.
- Functionality offered
  - build models,
  - apply models,
  - validate models,
  - access and query data in various ways.
- Technologies
  - REST style web services
  - RDF for description of resources
  - Links to existing and newly developed ontologies (mainly to describe metadata) about resources
Represenational State Transfer (REST)
A software architecture style, defined by Roy Fielding in his PhD thesis (2000). Many services worldwide offer REST API. There are (currently) no standards for RESTful applications, but merely design guides.

Design principles:

• Resource oriented
  – Every object (resource) is named and addressable (e.g. HTTP URL) Example: http://example.opentox.com/model/myBestModel, http://example.opentox.com/compound/50-00-0
  – RESTful API design starts by identifying most important objects and groups of objects, supported by the software system and proceeds by defining URL patterns.

• Transport protocol
  – HTTP is the most popular choice of transport protocol, but other protocols can be used as well

• Operations
  – All resources (nouns) support the same fixed and universal number of operations (verbs). HTTP (GET, POST, PUT, DELETE) operations are the common choice, when the transport protocol is HTTP.

• Hypermedia as the Engine of Application State
  – All resources should be reachable via a single (or minimum) number of entry points into RESTful applications. Thus, a representation of a resource should return hypermedia links to related resources.

• Error codes (for each resource/operation pair)
  – HTTP status codes (e.g. 200 OK, 400 Bad Request, 404 Not found, etc.) are usually used
OpenTox considers the following set of entities as essential building blocks:

- **Structures of chemical compounds**
- **Properties and identifiers** of chemical compounds
- **Datasets** of chemical compounds and various properties (measured or calculated)
- **Algorithms**
  - Data processing algorithms
    - Algorithms generating certain values, based on chemical structure (e.g. descriptor calculation)
    - Data preprocessing (e.g. Principal component analysis, feature selection)
    - Structure processing (e.g. structure optimization)
    - Algorithms, relating set of structures to another set of structures (e.g. similarity search or metabolite generation)
  - Machine learning algorithms
    - Supervised (e.g. Regression, Classification)
    - Unsupervised (e.g. Clustering)
  - Prediction algorithms, defined by experts (e.g. series of structural alerts, defined by human experts, not derived by learning algorithms)
• **Models** are generated by respective algorithms, given specific parameters
  • Statistical models are generated by applying statistical/machine learning algorithms to specific dataset and parameters
  • Models can be other than statistical, e.g. expert defined rules, quantum mechanical calculations, metabolite generation, etc. The intention of the framework is to be generic enough to accommodate varieties of predictive models.

• **Validation** provides procedures independent of model building facilities (e.g. crossvalidation) and generates relevant statistics.

• **Reports**
  – Various types of reports might be generated, using building blocks above (e.g. validation report can be generated using validation object, a model and a dataset).

• In addition, the following components are introduced:
  – **Task** (asynchronous processing of computationally intensive tasks)
  – **Authentication and authorization** (Ensuring secure access to sensitive resources)
  – **Ontology service** (provides an RDF storage and SPARQL endpoint for resources registration)
## Resources identification

All resources are identified via unique web address, assigned according to the URL templates

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>URL Template (example)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compound</td>
<td>Representations of chemical compounds</td>
<td><a href="http://host:port/compound/%7Bcompoundid%7D">http://host:port/compound/{compoundid}</a></td>
</tr>
<tr>
<td>Feature</td>
<td>Properties and identifiers</td>
<td><a href="http://host:port/feature/%7Bfeatureid%7D">http://host:port/feature/{featureid}</a></td>
</tr>
<tr>
<td>Dataset</td>
<td>Encapsulates set of chemical compounds and their property values</td>
<td><a href="http://host:port/dataset/%7Bdatasetid%7D">http://host:port/dataset/{datasetid}</a></td>
</tr>
<tr>
<td>Model</td>
<td>OpenTox model services</td>
<td><a href="http://host:port/model/%7Bmodelid%7D">http://host:port/model/{modelid}</a></td>
</tr>
<tr>
<td>Algorithm</td>
<td>OpenTox algorithm services</td>
<td><a href="http://host:port/algorithm/%7Balgorithmid%7D">http://host:port/algorithm/{algorithmid}</a></td>
</tr>
<tr>
<td>Task</td>
<td>Asynchronous jobs are handled via an intermediate Task resource. A resource, submitting an asynchronous job should return the URI of the task.</td>
<td><a href="http://host:port/task/%7Btaskid%7D">http://host:port/task/{taskid}</a></td>
</tr>
<tr>
<td>Ontology service</td>
<td>Provides storage and SPARQL search functionality for objects, defined in OpenTox services and relevant ontologies</td>
<td><a href="http://host:port/ontology">http://host:port/ontology</a></td>
</tr>
<tr>
<td>Authentication and authorisation</td>
<td>Granting access to protected resources for authorised users</td>
<td><a href="http://host:port/opensso">http://host:port/opensso</a> <a href="http://host:port/opensso-pol">http://host:port/opensso-pol</a></td>
</tr>
</tbody>
</table>
OpenTox REST operations

Individual resources (e.g. a dataset or a model)
- URI template http://host:port/{resource}/{resourceid}, e.g. http://host:port/model/{model_id} or http://host:port/dataset/{dataset_id}
- GET - retrieve representation of the resource
- PUT - update representation of the resource
- POST:
  - replace representation of the resource with a new one (e.g. replace the dataset with new content)
  - initiate calculations, based on this resource (e.g. submit dataset URI to an algorithm resource and obtain a model URI as a result)
- DELETE - delete the resource

Collections of resources (e.g. list of all available models, or datasets)
- URI template http://host:port/{resource}, (e.g. http://host:port/model or http://host:port/dataset)
- GET - retrieve representation of multiple resources (e.g. retrieve all available algorithms)
- PUT - N/A
- POST - create new resource and return its URI (e.g. create a new dataset by submitting new dataset content to the dataset service)
- DELETE - N/A
Build a predictive model

Create a model, Run calculations with dataset http://host1/dataset/id

Dataset service

Structures, descriptors, endpoints

HTTP POST

Regression, Classification, Quantum Chemistry, Descriptors, etc.

Algorithm service

Returns the model URL http://host1/model/id

Validation service

/validation/id

Model service

/model/{id}

HTTP POST

OpenTox

Published models, Algorithms, Ontologies, metadata

Ontology service
Uniform approach to models creation

Read data from a web address - process - write to a web address

Dataset
GET
POST
PUT
DELETE

Feature
GET
POST
PUT
DELETE

Compound
GET
POST
PUT
DELETE

Algorithm
GET
POST
PUT
DELETE

Model
GET
POST
PUT
DELETE

http://myhost.com/dataset/trainingset1

http://myhost.com/algorithm/neuralnetwork

http://myhost.com/model/predictivemodel1

OpenTox

Ideaconsult Ltd.
Use an algorithm to build a model

• An algorithm is applied by submitting HTTP POST to the algorithm URI and providing required parameters.
• A common required parameter is `dataset_uri=http://host:port/dataset/{datasetid}`, which specifies the data set to be operated on.
• HTTP POST in REST style services returns URI of the result, and not the content of the result.
• The algorithm services are designed to store the results into a dataset service and return the URL of the resulted dataset.
• In case of slow calculations a Task URI, instead of the dataset URI is returned.

```bash
```

* Connected to opentox.informatik.tu-muenchen.de (131.159.28.16) port 8080 (No)
  ➤ POST /OpenTox-dev/algorithm/J48 HTTP/1.1
  ➤ Host: opentox.informatik.tu-muenchen.de:8080
  ➤ Accept: */*
  ➤ Content-Type: application/x-www-form-urlencoded
  < HTTP/1.1 202 Accepted
  < Date: Sat, 31 Jul 2010 14:46:38 GMT
  < Location: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5
  < Accept-Ranges: bytes
  < Server: Noelios-Restlet-Engine/1.1.snapshot
  < Content-Type: text/uri-list;charset=ISO-8859-1
  < Content-Length: 99
  <
  * Connection #0 to host opentox.informatik.tu-muenchen.de left intact
  * Closing connection #0
  http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5
Resources: The model

• When task URI is returned, the returned status code is HTTP 202 Accepted, instead of HTTP 200 OK.
• This tells the client the processing is not completed and the client needs to poll the task URI until OK code is returned.
• The final result, returned by Example 25 is the URI of the new model http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48.
• To obtain prediction results, POST a dataset to the model URI $ curl -iv -H "Accept:text/uri-list" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5

> GET /OpenTox-dev/task/acdf6eac-d5a2-402c-a4e2-06cd7e3ca1b5 HTTP/1.1
> User-Agent: curl/7.18.2 (x86_64-pc-linux-gnu) libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.5 libidn/1.8 libssh2/0.18
> Host: opentox.informatik.tu-muenchen.de:8080
> Accept:text/uri-list
>
< HTTP/1.1 200 OK
< Date: Sat, 31 Jul 2010 14:47:22 GMT
Date: Sat, 31 Jul 2010 14:47:22 GMT
< Location: http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48
< Vary: Accept-Charset, Accept-Encoding, Accept-Language, Accept
< Accept-Ranges: bytes
< Server: Noelios-Restlet-Engine/1.1.snapshot
< Content-Type: text/uri-list; charset=ISO-8859-1
< Content-Length: 86
<
* Connection #0 to host opentox.informatik.tu-muenchen.de left intact
* Closing connection #0
http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_j48_48
Uniform approach to data processing (e.g. Descriptors calculation)

Read data from a web address - process - write to a web address

Dataset
GET
POST
PUT
DELETE

Feature
GET
POST
PUT
DELETE

Compound
GET
POST
PUT
DELETE

Algorithm
GET
POST
PUT
DELETE

Dataset
GET
POST
PUT
DELETE

Feature
GET
POST
PUT
DELETE

Compound
GET
POST
PUT
DELETE

= http://myhost.com/dataset/trainingset1

+ http://myhost.com/algorithm/{descriptorX}

= http://myhost.com/dataset/results

OpenTox

Ideaconsult Ltd.
Uniform approach to models validation and report generation

Read data from a web address - process - write to a web address

Validation report

Validation
GET
POST
PUT
DELETE

Dataset
GET
POST
PUT
DELETE

Model generating predictions
Model
GET
POST
PUT
DELETE

Report
GET
POST
PUT
DELETE

http://myhost.com/dataset/trainingset1
http://myhost.com/dataset/predictedresults1
http://myhost.com/validation
http://myhost.com/model/predictivemodel1

http://myhost.com/report/1
Apply predictive models

/model/{id}

HTTP POST

Apply the model
http://host1/model/id

to dataset
http://host2/dataset/id

HTTP POST

SPARQL

Ontology service

Published models,
Algorithms,
Ontologies,
metadata

Retrieves available
endpoints and model
URLs, e.g.
http://host1/model/id

Returns the results
dataset URL
http://host/dataset/id

Model service
Uniform approach to model prediction

Read data from a web address - process - write to a web address

- Dataset
  - GET
  - POST
  - PUT
  - DELETE

- Feature
  - GET
  - POST
  - PUT
  - DELETE

- Compound
  - GET
  - POST
  - PUT
  - DELETE

- Model
  - GET
  - POST
  - PUT
  - DELETE

- Dataset
  - GET
  - POST
  - PUT
  - DELETE

- Feature
  - GET
  - POST
  - PUT
  - DELETE

- Compound
  - GET
  - POST
  - PUT
  - DELETE

http://myhost.com/model/predictivemodel1

http://myhost.com/dataset/id1

http://myhost.com/dataset/results1
Uniform access to data

Dataset
- GET
- POST
- PUT
- DELETE

Feature
- GET
- POST
- PUT
- DELETE

Compound
- GET
- POST
- PUT
- DELETE

Dataset service
- Structures, endpoints & predictions
- Published models, Algorithms, Ontologies, metadata

Ontology service
- Feature
  - GET
  - POST
  - PUT
  - DELETE
- Compound
  - GET
  - POST
  - PUT
  - DELETE

HTTP GET / POST

Find chemical compounds, return dataset URL; http://host2/compound/id

Upload data, receive dataset URL http://host2/dataset/id

OpenTox

Published your data, retrieve linked data.
RDF - Resources representation

- The `opentox.owl` ontology
  - A common OWL data model of all OpenTox resources
  - Describes OpenTox resources
  - Describes relationships between them
  - Generates object's RDF representations.
- RDF/XML representation is mandatory for OpenTox resources.
- Uniform approach to data representation
  - Calculated and measured properties of chemical compounds are represented in an uniform way
  - Linked to the resource used for data generation
  - Annotated via ontology entries
  - Model representations link to algorithms and data used

All OpenTox components are defined by OWL ontology

http://opentox.org/api/1.1/opentox.owl

All resources are subclasses of `ot:OpenToxResource`
## Resources: Chemical compound

### Compound
Provides different representations for chemical compounds with a unique and defined chemical structure.

```
/compound/{id}
```

### Conformer
```
/compound/{id}/conformer/{id}
```

### Documentation
http://opentox.org/dev/apis/api-1.1/structure

### Representation
A subclass of ot:OpenToxResource. Supports different Chemical MIME formats
- RDF representation only for specifying owl:sameAs links to external resources

---

### Example 1. Retrieve compound as MOL

```
$ curl -H "Accept:chemical/x-mdl-molfile"
http://apps.ideaconsult.net:8080/ambit2/compound/1
```

CH2O

```
APtclactv09040902283D 0 0.00000 0.00000
4 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-0.6004 0.0000 0.0001 O 0 0 0 0 0 0 0 0 0 0 0
0.6072 0.0000 -0.0004 C 0 0 0 0 0 0 0 0 0 0 0
1.1472 0.9353 0.0016 H 0 0 0 0 0 0 0 0 0 0 0
1.1472 -0.9353 0.0016 H 0 0 0 0 0 0 0 0 0 0 0
1 2 2 0 0 0 0
2 3 1 0 0 0 0
2 4 1 0 0 0 0
```

### Example 2. Retrieve compound as SMILES

```
$ curl -H "Accept:chemical/x-daylight-smiles"
http://apps.ideaconsult.net:8080/ambit2/compound/1
```

O=C

### Example 3. Query compounds

```
$ curl -H Accept:chemical-mime "
http://apps.ideaconsult.net:8080/ambit2/query/compound/{any-identifier-or-keyword}
```

```
$ curl -H Accept:chemical-mime "
http://apps.ideaconsult.net:8080/ambit2/query/smarts?search={smarts}
```
Feature
A Feature is a resource, representing any kind of a property or identifier, assigned to a Compound.
The feature types are determined via their links to ontologies (Feature ontologies, Descriptor ontologies, Endpoints ontologies).
/feature/{id}

Documentation
http://opentox.org/dev/apis/api-1.1/feature

Representation
ot:Feature, a subclass of ot:OpenToxResource.
Mandatory RDF/XML format

Properties
• Name, defined by dc:title (Dublin Core namespace);
• Units, defined by ot:units annotation property (OpenTox namespace);
• Creator, defined by dc:creator annotation property (Dublin Core namespace);
• The origin of the Feature is defined by ot:hasSource object property (OpenTox namespace) element and can be ot:Algorithm, ot:Model or ot:Dataset;
• Relations to other resources, which represent the same entity, could be established via owl:sameAs property. This approach can be used for example to link the ot:Feature resource to a resource from another ontology (an example follows)
• There are subclasses of ot:Feature (namely), which are used ot:NumericFeature, ot:StringFeature, ot:NominalFeature denote if a feature holds numeric, nominal or string values.
The example shows an OpenTox feature with title “XLogP” and identified by the URI http://apps.ideaconsult.net:8080/ambit2/feature/22114
• Linked to an entry of a simplified ontology of toxicological endpoints

The algorithm used to generate values for this feature
• Specified by ot:algorithm property
• Identified by the URI http://apps.ideaconsult.net:8080/ambit2/algorithm/org.openscience.cdk.qsar.descriptors.molecular.XLogPDescriptor

• Note the URI identifies an OpenTox Algorithm resource
• The algorithm URI itself is dereferensable
• Can be used to initiate calculations of XLogP descriptor.

```bash
```

```
<rdf:RDF
  xmlns:ot="http://www.opentox.org/api/1.1#"
  xmlns:otee="http://www.opentox.org/echaEndpoints.owl#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns="http://apps.ideaconsult.net:8080/ambit2/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://apps.ideaconsult.net:8080/ambit2/"
>
  <owl:Class rdf:about="http://www.opentox.org/api/1.1#Algorithm"/>
  <owl:Class rdf:about="http://www.opentox.org/api/1.1#Feature"/>
  <owl:Class rdf:about="http://www.opentox.org/api/1.1#NumericFeature"
    rdf:subClassOf rdf:resource="http://www.opentox.org/api/1.1#Feature"/>
  <owl:Class>
    <ot:NumericFeature rdf:about="feature/22114">
      <ot:hasSource>
        <ot:Algorithm rdf:about="algorithm/org.openscience.cdk.qsar.descriptors.molecular.XLogPDescriptor"/>
      </ot:hasSource>
      <owl:sameAs rdf:resource="otee:Octanol-water_partition_coefficient_Kow"/>
      <dc:title>XLogP</dc:title>
    </ot:NumericFeature>
  </owl:Class>
</rdf:RDF>
```
Resources: Feature (an example)

An example (N3) of another ot:Feature resource, XLogP descriptor (again)

- Generated by different implementation.
- In this case, its name is “TUM_CDK_XLogP”, and the algorithm resource used to generate resides at Technical University of Munich (TUM) premises http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/CDKPhysChem/XLogPDescriptor. This algorithm URL could also be used to initiate descriptor calculations.

- The representation of Algorithm resources refers to the BlueObelisk ontology entry http://www.blueobelisk.org/ontologies/chemoinformatics-algorithms/#xlogP

```
@prefix ot: <http://www.opentox.org/api/1.1#> .
@prefix dc: <http://purl.org/dc/elements/1.1#> .
@prefix : <http://apps.ideaconsult.net:8080/ambit2/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix af: <http://apps.ideaconsult.net:8080/ambit2/feature/> .
ot:hasSource
  a owl:ObjectProperty .
ot:units
  a owl:DatatypeProperty .
ot:Feature
  a owl:Class .
ot:NumericFeature
  a owl:Class ;
rdfs:subClassOf ot:Feature .
af:26184
  a ot:Feature , ot:NumericFeature ;
dc:creator "http://ambit.uni-plovdiv.bg:8080/ambit2" ;
dc:title "TUM_CDK_XLogP" ;
ot:hasSource <http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/CDKPhysChem/XLogPDescriptor> ;
  ot:units "" ;
  = otee:Octanol-water_partition_coefficient_Kow .
```
Toxicity endpoints ontologies

- Derived from ECHA classification of endpoints, published in REACH guidance documents
- Physicochemical properties and various toxicological endpoints
- The hierarchy doesn’t represent the complexity of toxicological assays, but can be used as a first approximation to assign meaning to the data entries and generate REACH report.
- More specific description of toxicological assays can be used as well.
- Ontologies for specific toxicity assays are developed by OpenTox partners

The ECHA endpoints ontology
http://www.opentox.org/echaEndpoints.owl
Resources: a feature

An illustration of **ot:Feature**, imported from a file and not calculated.
The example shows a feature, representing EINECS number, imported from the ECHA preregistration list.

```
curl -H "Accept:application/rdf+xml"
http://apps.ideaconsult.net:8080/ambit2/feature/3
<rdf:RDF
   xmlns:ot=http://www.opentox.org/api/1.1#
   xmlns="http://apps.ideaconsult.net:8080/ambit2/"
   xml:base="http://apps.ideaconsult.net:8080/ambit2/">
   <ot:Feature rdf:about="feature/3">
      <dc:creator>
         <ot:hasSource>ECHA...</ot:hasSource>
         <owl:sameAs rdf:resource="http://www.opentox.org/api/1.1#EINECS"/>
         <ot:units></ot:units>
         <dc:title>EC</dc:title>
      </ot:Feature>
   </rdf:RDF>
```
Feature: summary

- OpenTox Feature resource uniquely identifies properties and identifiers, assigned to a compound, via feature URIs.
- These URIs are dereferencable
- Allow to assign different levels of meaning, by linking to entries to ontologies (e.g. algorithms or toxicological endpoints), as well as linking to the algorithms, which can be used to generated property values.
- The same approach can be used to denote assays, provided that the assay is defined by an ontology, species, functional groups, etc.
Resources: Dataset

Dataset
Provides access to chemical compounds and their features (e.g. structural, physical-chemical, biological, toxicological properties)

Operations
POST – Upload a dataset
PUT – Update the dataset content
DELETE – Remove the dataset

Representation
RDF/XML (mandatory)
• The dataset consists of data entries.
• Each entry is associated with exactly one chemical compound, identified by its URI and available via OpenTox Compound service API;
• One and the same compound can be associated with multiple dataset entries;
• Every “column” is associated with a Feature, its representation should be available via OpenTox Feature API
Resources: Dataset

The dataset services optionally support formats other than RDF

- text/csv (Comma delimited),
- text/x-arff (Weka ARFF),
- application/pdf,
- chemical/x-mdl-sdfile,
- other Chemical MIME formats

This allows retrieving the same data in convenient format, but the URL links to compound and feature resources are being lost.
## Dataset: metadata and features

<table>
<thead>
<tr>
<th>Description</th>
<th>URI Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve entire dataset content. If uri-list, retrieve only compound URIs</td>
<td><a href="http://host:port/dataset/%7Bid%7D">http://host:port/dataset/{id}</a></td>
</tr>
<tr>
<td>Retrieve representation of features (columns) of the dataset</td>
<td><a href="http://host:port/dataset/%7Bid%7D/feature">http://host:port/dataset/{id}/feature</a></td>
</tr>
<tr>
<td>Retrieves dataset metadata (name, etc.)</td>
<td><a href="http://host:port/dataset/%7Bid%7D/metadata">http://host:port/dataset/{id}/metadata</a></td>
</tr>
</tbody>
</table>

```
<rdf:RDF xmlns:ot="http://www.opentox.org/api/1.1#"
    ...
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:base="http://apps.ideaconsult.net:8080/ambit2/"
<ot:Dataset rdf:about="dataset/9">
    <dc:source>ISSCAN_v3a_1153_19Sept08.1222179139.sdf</dc:source>
    <dc:publisher>somebody</dc:publisher>
    <rdfs:seeAlso>
        <bx:Entry rdf:about="reference/20117">
            <rdfs:seeAlso>http://www.epa.gov/NCCT/dsstox/sdf_isscan_external.html</rdfs:seeAlso>
            <dc:title>ISSCAN_v3a_1153_19Sept08.1222179139.sdf</dc:title>
        </bx:Entry>
    </rdfs:seeAlso>
    <dc:title>ISSCAN: Istituto Superiore di Sanita, CHEMICAL CARCINOGENS: STRUCTURES AND EXPERIMENTAL DATA</dc:title>
</ot:Dataset>
</rdf:RDF>
```
Data publishing

1) POST a file with chemical structures and properties to OpenTox dataset service.
   • The structures and data are assigned a dataset URL and become available by multiple formats (RDF, Chemical MIME, CSV, Weka ARFF)

2) Assign metadata
   • PUT /dataset/{id}/metadata

3) Annotate any of dataset features /dataset/{id}/feature by assigning links to relevant ontologies
   • PUT /feature/{id}

Find chemical compounds, return dataset URL; http://host2/compound/id
Upload data, receive dataset URL http://host2/dataset/id

HTTP GET / POST

Dataset service

Annotation

Algorithms ontologies

Toxicology related ontologies
Resources: Algorithm

Algorithm
Provides access to OpenTox algorithms. There are several algorithm services, developed by different OpenTox partners. List of algorithms can be retrieved by HTTP GET operation at http://host:port/algorithm

```
curl -H "Accept:text/uri-list" http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd
```

- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/kNNclassification
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/J48
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/kNNregression
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/PLSregression
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/M5P
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/GaussP
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/FTM/{smiles}
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/gSpan/{smiles}
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/CDKPhysChem
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/JOELIB2
- http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithmd/InfoGainAttributeEval
Resources: Algorithm

Representation

- Multiple type of algorithms:
  - descriptor calculation algorithms,
  - machine learning procedures
  - data preprocessing.
- The representation of algorithms is again defined by Opentox ontology, where all algorithms are subclass of ot:Algorithm

Algorithm types ontology

http://opentox.org/data/documents/development/RDF%20files/AlgorithmTypes

- provides a classification of algorithm types.
- Algorithm type in RDF representation is set by direct subclassing (rdf:type) of a class from the algorithm types ontology (ota: http://www.opentox.org/algorithms.owl ), e.g. <myalgorithm> rdf:type ota:Classification.
Resources: Algorithm

Representation
• Algorithm name is defined by dc:title
Parameters, supported by the algorithm are specified via object property ot:parameters and should be of class ot:Parameter (as defined in opentox.owl).
These entries serve as a information what parameters are required in order to run the algorithm, the values itself should be provided by the client when initiating the calculations via POST.
• Algorithm types are distinguished by means of Algorithm types ontology.

$ curl -H "Accept:application/rdf+xml"
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ot="http://www.opentox.org/api/1.1#"
  xmlns:ota=http://www.opentox.org/algorithmTypes.owl#
  <ota:Supervised
  <dc:dc:title rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Classification: Decision tree J48</dc:title>
  <rdf:type>
    rdf:resource="http://www.opentox.org/algorithmTypes.owl#Classification"/>
  <dc:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></dc:description>
  <rdf:type>
    rdf:resource="http://www.opentox.org/algorithmTypes.owl#SingleTarget"/>
  <dc:dc:pubDate rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >Somebody</dc:dc:pubDate>
  <rdf:type>
    rdf:resource="http://www.opentox.org/algorithmTypes.owl#EagerLearning"/>
  </rdf:type>
</ota:Supervised>
</rdf:RDF>
Resources: Model

- Representations of predictive models.
- A Model is created by HTTP POST to an ot:Algorithm with specific parameters and/or input ot:Dataset.

**Representation**
- Model Name is defined by dc:title property
- Model creator might be defined by dc:creator property
- The date of Model creation is defined by dc:date property
- The Algorithm defined by ot:algorithm object property
- The independent variables are instances of ot:Feature defined by ot:independentVariables property (can be multiple)
- The dependent variables are are instances of ot:Feature and are defined by ot:dependentVariables property (can be multiple)
- The variables, where prediction results will be stored, are are instances of ot:Feature and are defined by ot:predictedVariables property (can be multiple)
- Parameters are defined by ot:parameters
- The training Dataset is an instance of ot:Dataset and defined by ot:trainingDataset
Linked resources: Compound, Algorithm, Model, Dataset, Features
Linked resources: Compound, Algorithm, Model, Dataset, Features

Dataset Resource

Descriptor resource

Assay resource

Chemical compound

Blue Obelisk algorithms ontology

Regression Classification Quantum Chemistry Descriptors, etc.

OpenTox algorithm types ontology

Toxicology related ontologies
Register at OpenTox ontology service
- RDF triple storage
- Accepts HTTP POST
- SPARQL endpoint

Curl -X POST -d
"uri=http://apps.ideaconsult.net:8080/ambit2/model/57"
http://apps.ideaconsult.net:8080/ontology

Becomes visible for applications
All components are implemented as REST web services. There could be multiple implementations of same type of components. (Subset of) services could be hosted by the same provider, or by multiple providers on separate locations.

<table>
<thead>
<tr>
<th>Partner No./Service type</th>
<th>Compound</th>
<th>Dataset</th>
<th>Feature</th>
<th>Algorithm (processing)</th>
<th>Algorithm (model)</th>
<th>Model</th>
<th>Task</th>
<th>Report</th>
<th>Validation</th>
<th>Authentication and Authorisation service</th>
<th>Ontology service</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 (IDEA)</td>
<td></td>
<td></td>
<td></td>
<td><a href="http://apps.ideaconsult.net:8080/ambit2">http://apps.ideaconsult.net:8080/ambit2</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OpenTox services can be used to develop specific applications, or embedded in workflow systems.

### Framework
- Toxicity data
- Predictive models
- Validation support
- Interpretation aids

### Unified Access
- Toxicologists
- Modelers
- API for new algorithms development & integration

### Open Source
- To optimise impact
- To allow inspection / review
- To attract external contributors
Demo applications

- Two end user oriented demo applications, making use of OpenTox webservices, have been developed, deployed and are available for testing - [http://toxcreate.org](http://toxcreate.org) and [http://toxpredict.org](http://toxpredict.org);
- ToxCreate creates models from user supplied datasets;
- ToxPredict uses existing OpenTox models to estimate chemical compound properties
RDF: Lessons learned

• OpenTox specific
  – it hasn’t started as Linked data/RDF project!
• REST and RDF mix is not (yet?) popular
  … but is natural to be able to retrieve (partial) resource representation, described by triples
• Steep learning curve
• Some hard topics:
  – Data model vs. format
  – The subject-predicate-object concept vs. tabular/hierarchical/other implicit structure
  – The recognition of the added value?
    • XML, JSON, YAML, plain text etc. vs. RDF
RDF: Performance

RDF representation is verbose ...

... and in-memory RDF libraries are slow ...

A dataset with 320 chemicals, 60 columns

A dataset with 6500 chemicals, 12 columns

... lack of streaming parsers/writers ...
RDF: Wish list

• Convenient explanation of subject-predicate-object concept for beginners
• A (high performant) triple storage should not be a mandatory requirement to publish RDF data
• Fast streaming parsers and writers
• (terse) JSON serialisation
• Security
• Synchronisation of distributed RDF content
Thank you!

Build an application with OpenTox REST Web Services API
http://opentox.org/dev/apis/api-1.1

Download AMBIT Implementation of OpenTox API and launch your OpenTox service
http://ambit.sourceforge.net