



Deliverable D2.2

Report on the Initial Framework Prototype

Grant Agreement	Health-F5-2008-200787
Acronym	OpenTox
Name	An Open Source Predictive Toxicology Framework
Coordinator	Douglas Connect

Contract No.	Health-F5-2008-200787	
Document Type:	Deliverable Report	
WP/Task:	WP2	
Name	Report on Prototype Framework	
Document ID:	OpenTox Deliverable Report D2.2	
Date:	March 8, 2010	
Status:	Final, v1.9	
Organisation:	Albert-Ludwigs Universität, Freiburg	
Contributors	Andreas Karwath	ALU-FR
	Martin Gütlein	ALU-FR
	Nina Jeliaskova	IDEA
	Barry Hardy	DC
	Nicki Douglas	DC
	Andreas Maunz	IST

Distribution:	Public
---------------	--------

Purpose of Document:	To document results for this deliverable
----------------------	--

Document History:	1 - Initial draft prepared on Feb 15, 2010
	2 - Updated Draft, 25 Feb, 2010
	3 - Updated Draft, 26 Feb, 2010
	4 - Updated Draft, 27 Feb, 2010
	5 - Final Version approved, 28 Feb 2010
	6 - Updated Version, 8 March 2010

Table of Contents

Summary	5
1. Introduction	7
1.1 OECD Principles of (Q)SAR Validation	7
1.1.1 Defined Endpoint	7
1.1.2 An Unambiguous Algorithm.....	9
1.1.3 Defined Applicability Domain	10
1.1.4 Appropriate Measures of Goodness-of-Fit, Robustness and Predictivity	10
1.1.5 A Mechanistic Interpretation, if possible	11
1.2 OpenTox Design Principles	12
1.2.1 Interoperability	12
1.2.2 Flexibility	12
1.2.3 Transparency	12
1.2.4 Extensibility	13
2. Implementation Principles	13
2.1 Open Source.....	14
2.2 Open and Distributed Access	14
2.3 Open Interfaces.....	15
3. OpenTox API	15
3.1 Overview	15
3.2 Web services	16
3.2.1 Compound.....	17
3.2.1.1 Conformers (optional)	18
3.2.2 Feature.....	19
3.2.3 Dataset	20
3.2.4 Algorithm.....	22
3.2.5 Model.....	22
3.2.6 Validation	23
3.2.6.1 Standard Validation	23

3.2.6.2	Cross-Validation.....	25
3.2.6.3	Validation – Report.....	26
3.2.7	Task.....	27
3.2.8	Ontology Service.....	28
4.	Prototype Use Cases	29
4.1	ToxPredict Use Case	29
4.1.1	Interaction of OpenTox services, employed in ToxPredict	36
4.2	ToxCreate Use Case	39
4.3	Validation Use Case: Building and Validating a Model	44
4.4	Testing Results	47
4.4.1	Server Testing.....	48
4.4.2	API testing	49
4.4.3	Use case testing.....	50
5.	Discussion	54
5.1	Further Working Directions	54
5.2	Conclusions	55
6.	Appendix	57
1	General Instructions.....	57
2	Beta Testing Objectives	57
3	Beta Testing Tasks.....	57
4	Known ToxPredict Problems	58
5	Part-A: Identification.....	59
6	Part-B: Functional Evaluation	59
7	Part-C: Overall Comments and Usability Evaluation.....	60
8	Part-D: Specific Bugs and Problems Noted	62
9	Part-E: Other Generic Topics	62

Summary

The **OpenTox Framework** consists of a set of distributed web services for the construction and application of predictive toxicity models. The Framework includes services for compounds, datasets, features, algorithms, models, ontologies, tasks, validation, and reporting which may be combined into multiple applications satisfying a variety of user needs. The guiding principles in the construction of OpenTox applications are based on the OECD Principles of (Q)SAR Validation¹, satisfying **REACH** legislation and user requirements, and the additional design principles of interoperability, flexibility, transparency and extensibility. A key feature of the OpenTox Framework is that it has been designed in a multi-domain friendly way, which is essential for data and model sharing, reproducibility and validation of prediction results. We report on how these principles influenced the design and construction of the OpenTox Framework. The OpenTox Application Programming Interfaces which connect multiple distributed web services in an interoperable manner are described in detail. Based on these web services, two user applications were created: a) **ToxPredict** which predicts and reports on toxicities for endpoints for a user-provided input chemical structure, and b) **ToxCreate** which builds and validates a predictive toxicity model based on a user-provided input toxicology dataset. The results of initial user testing of both these applications are presented, and the issues and lessons learned for subsequent development discussed.

The **OpenTox Framework** supports rapid application development and extensibility by using well-defined ontologies, allowing simplified communication of data and meaning between individual services. The ToxCreate and ToxPredict applications show the potential impact of the Framework regarding high-quality and consistent structure-activity relationship modeling of REACH-relevant endpoints. The applications have been made available publicly on the Web (www.opentox.org/toxicity-prediction) providing immediate user access to the applications as they have been developed. User-based testing and reporting provides a mechanism for users to provide feedback on features and requests which can be quickly taken into account in the agile development approach pursued, so as to improve the services offered to users in a timely manner.

ToxPredict satisfies a common and important situation for a user wishing to evaluate the toxicity of a chemical structure. The user may upload or draw the chemical structure in a web browser and quickly obtain a report back on what current data and model predictions are available for the toxicity endpoints they have interest in. The user does not have to

¹ ecb.jrc.ec.europa.eu/qsar/background/index.php?c=OECD

cope with many current challenges such as the difficulty of finding or using existing data or creating and using complicated computer models. Because of the extensible nature of the standardised design of the OpenTox Framework, many new datasets and models from other researchers may be easily incorporated in the future, both strengthening the value offered to the user and ensuring that research results are not left languishing unused in some isolated, unintegrated resource not accessible to the user. The approach offers the potential to be extended to the complete and easy-to-use generation of reporting information on all REACH-relevant endpoints based on existing available scientific research results, and indications when additional experimental work is required, thus satisfying currently unmet industry and regulatory needs.

ToxCreate provides a resource to modellers to build soundly-based predictive toxicology models, basely solely on a user-provided input toxicology dataset that can be uploaded through a web browser. The models can be built and validated in an automated and scientifically sound manner, so as to ensure that the predictive capabilities and limitations of the models can be examined and understood clearly. Models can subsequently be easily made available to other researchers and combined seamlessly into other applications through the OpenTox Framework. Barriers of interoperability between applications and content that are current significant pain points of cost and time for industry users are removed, as the user may combine data, models and validation from multiple sources in a dependable and time-effective way.

1. Introduction

The OpenTox Framework prototype consists of a set of distributed web services for the construction and application of predictive toxicity models. The Framework includes services for datasets, algorithms, features, models, validation, and reporting. The guiding principles in the construction of the prototype are the OECD Principles of (Q)SAR Validation, and the additional design and application principles of interoperability, flexibility, transparency and extensibility. On a technical basis, the construction of the framework was guided by open source development, incorporation of standards and ontologies, and distributed integration of web services into applications, enabling participation from multiple resource providers. In the sub-sections below we will discuss these principles and how they were incorporated into the prototype.

1.1 OECD Principles of (Q)SAR Validation

Here we briefly review the five OECD (Q)SAR validation principles and their relevance to the OpenTox prototype development. We have incorporated these principles in the OpenTox Framework design as much as possible and wherever appropriate.

1.1.1 Defined Endpoint

(Q)SAR model quality crucially depends on the clarity of endpoints and experimental protocols used and the ability to communicate this information in an unambiguous way, both in model development and model application. The current practice usually includes a textual description of the materials and methods used for acquiring experimental data as well as literature references, while the model description is a separate entity. The challenge to the distributed web services framework, described in this report, was to provide an automatic and unique way of describing and linking the endpoint information in a formal way, able to be processed automatically by the software, with minimal human interaction. This is currently solved by making use of a simple ontology of endpoints. We have defined an ontology based on the OWL (Web Ontology Language)² for toxicological endpoints which is in line with current ECHA REACH guidance³. Using this ontology, each attribute in a toxicological dataset can be associated with an entry to the ontology, therefore allowing a unique mapping between endpoints in various and heterogeneous datasets. This

² www.w3.org/TR/owl-features/

³ guidance.echa.europa.eu/docs/guidance_document/information_requirements_r6_en.pdf?vers=20_08_08

ontology possesses 5 subclasses: ecotoxic effects, environmental fate parameters, human health effects, physio-chemical effects, and toxicokinetics. Each of these subclasses has one or two further layers of subclasses. A graphical overview can be seen in Figure 1.1 whereas Figure 1.2 shows a level 3 subclass for *carcinogenicity*. In the future, this ontology will be extended to include complete information about the test study of the dataset.

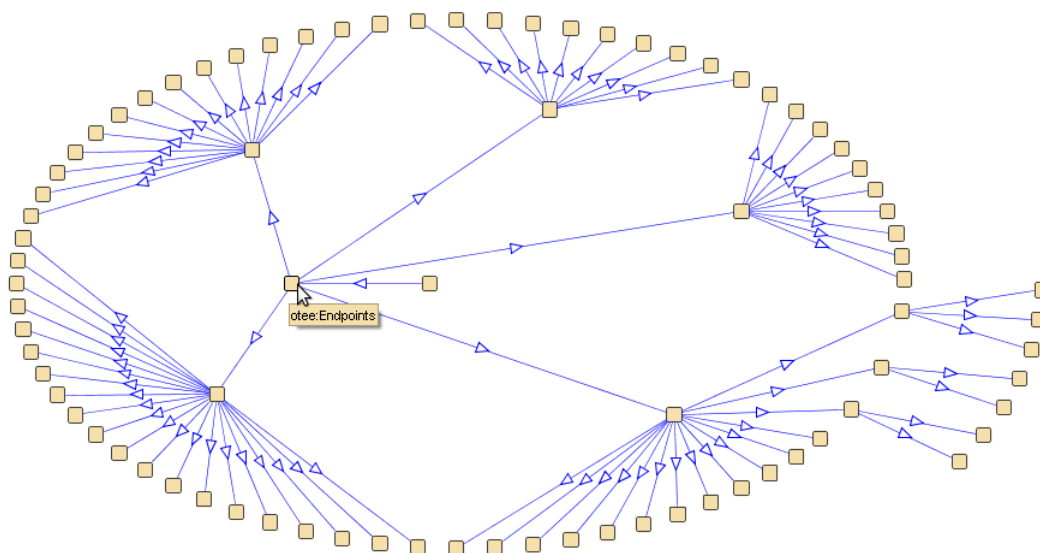


Figure 1.1 A graphical overview of the ECHA endpoints ontology in OWL

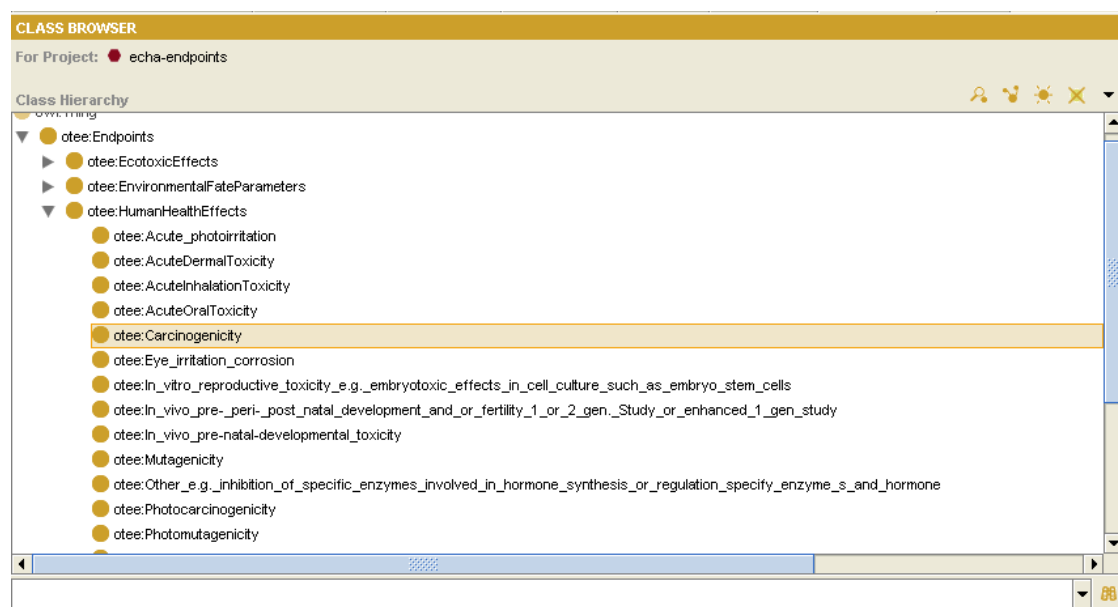


Figure 1.2 The carcinogenicity subclass in the human health effect class of the ontology

The endpoint ontology can be accessed in the development documents section of the OpenTox website⁴ and can be viewed with the Protégé⁵ editor.

1.1.2 An Unambiguous Algorithm

Currently OpenTox is deploying an algorithm type ontology⁶. This ontology allows a clear definition of what type of algorithm is used to construct a model. Figure 1.3 shows a graphical overview of the current version of this ontology. The plan is to extend this ontology in the future to a full description of every algorithm, including references, parameters and default values. This will be achieved by adopting the Blue Obelisk ontology⁷ and is currently work-in-progress.

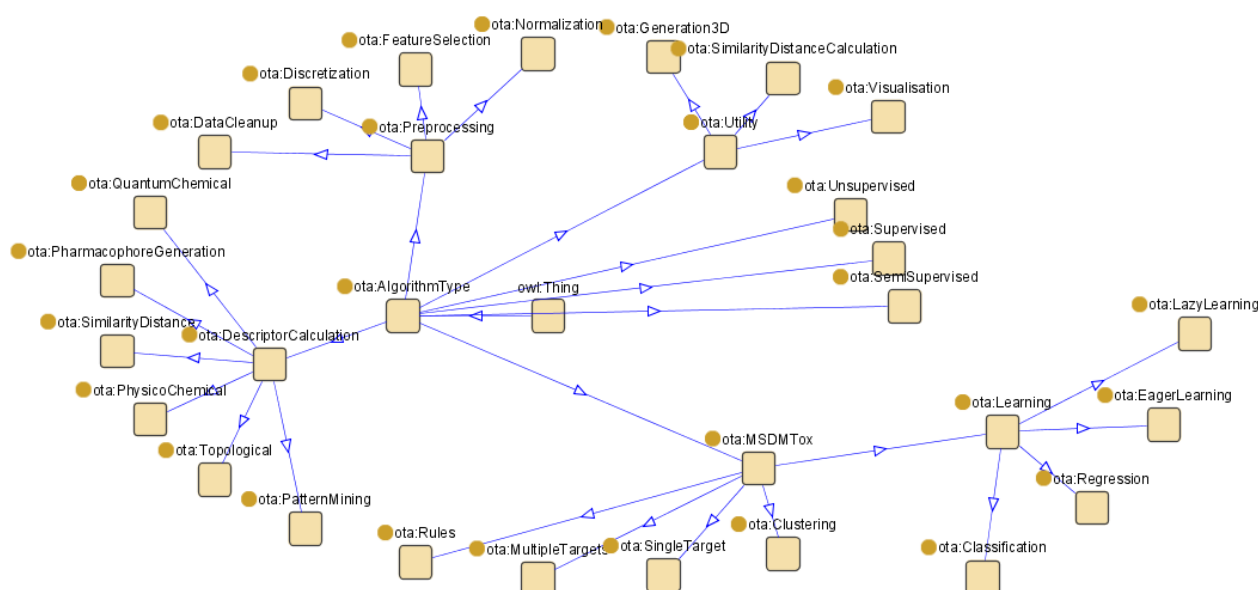


Figure 1.3 A graphical overview of the current AlgorithmType ontology

⁴ [www.opentox.org/data/documents/development/RDF files/Endpoints/](http://www.opentox.org/data/documents/development/RDF%20files/Endpoints/)

⁵ protege.stanford.edu

⁶ [www.opentox.org/data/documents/development/RDF files/AlgorithmType](http://www.opentox.org/data/documents/development/RDF%20files/AlgorithmType)

⁷ qsar.svn.sf.net/viewvc/qsar/trunk/qsar-dicts/descriptor-ontology.owl?revision=218

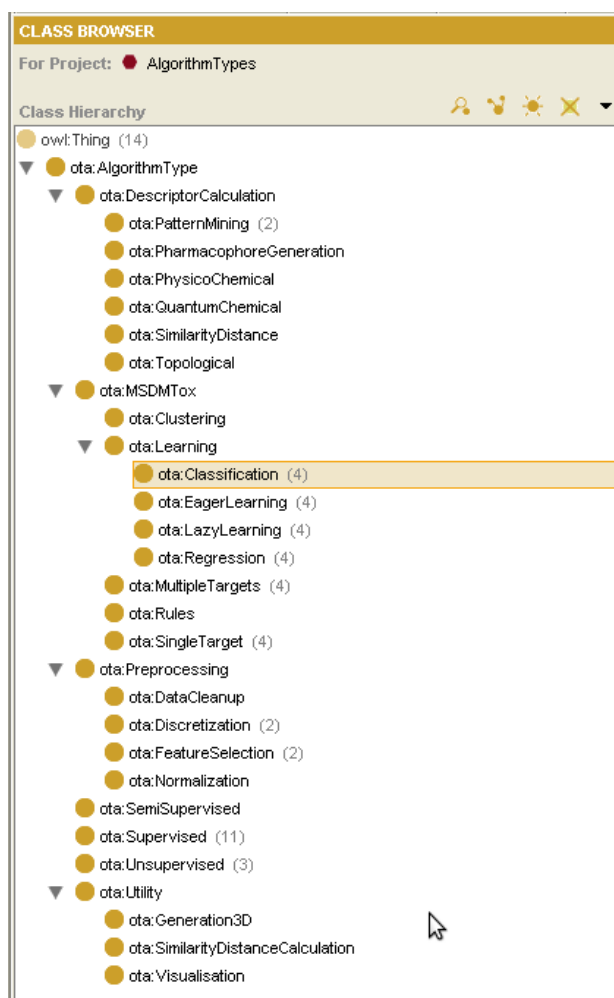


Figure 1.4 A textual overview of the algorithm type ontology.

1.1.3 Defined Applicability Domain

We handle applicability domain as an algorithm or model, e.g. a specific applicability domain algorithm is applied to a dataset, and the result is then an applicability domain model. This model can then be applied to reason about the applicability of a model when applied to a novel compound. Currently, this approach is not yet fully reflected within the ontology. Nevertheless, we are planning to integrate this in forthcoming ontology updates, as well as in the next development iteration of the API (1.2), which is scheduled to be completed for September 2010.

1.1.4 Appropriate Measures of Goodness-of-Fit, Robustness and Predictivity

Within the validation part of the prototype framework, we have concentrated so far on including validation and cross-validation objects. These include a wide variety of measures

for evaluating the quality of models generated by algorithms on the datasets. These measures include for classification tasks:

Name	Explanation
Confusion Matrix	A confusion matrix is a matrix, where each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two or more classes.
Absolute number and percentage of unpredicted compounds	Some compounds might fall outside the applicability domain of the algorithm or model. These numbers provide an overview on the applicability domain fit for the compound set requiring prediction.
Precision, recall, and F2-measure	These three measures give an overview on how pure and how sensitive the model is. The F2-measure combines the other two measures.
ROC curve plot and AUC	A receiver operating characteristic (ROC) curve is a graphical plot of the true-positive rate against the false-positive rate as its discrimination threshold is varied. This gives a good understanding of how well a model is performing. As a summarisation performance scalar metric, the area under curve (AUC) is calculated from the ROC curve. A perfect model would have area 1.0, while a random one would have area 0.5.

And for regression tasks:

Name	Explanation
MSE and RMSE	The mean square error (MSE) and root mean squared error (RMSE) of a regression model are popular ways to quantify the difference between the predictor and the true value.
R^2	The explained variance (R^2) provides a measure of how well future outcomes are likely to be predicted by the model. It compares the explained variance (variance of the model's predictions) with the total variance (of the data).

1.1.5 A Mechanistic Interpretation, if possible

As mechanistic interpretation often relies on human knowledge, this usually cannot be done automatically. However, in the current API it is foreseen to generate skeletons for reporting using the validation results created by extensive testing during model construction, allowing subsequent user-entered explanations about mechanisms. Other potential future extensions of OpenTox services could include resources providing insight on mechanisms, e.g. from pathways and systems biology models, selection and inclusion of *in vitro* assays relevant to the mechanism in the model, or from data mining of human adverse events data.

QMRf reporting will be facilitated by integration of the existing QMRf editor⁸ into OpenTox applications, and allowing the end-users to annotate models with the information required by the QMRf format.

1.2 OpenTox Design Principles

The design principles of interoperability, flexibility, transparency and extensibility are key ingredients of the OpenTox Framework design, which additionally guide its architecture and implementation.

1.2.1 Interoperability

Interoperability with respect to the OpenTox Framework refers to the principle that different OpenTox components or services may correctly exchange information with each other and subsequently make use of that information. Both syntactic interoperability for correct data exchange and semantic interoperability supporting the accurate communication of meaning and interpretation of data are supported principles for OpenTox resources. The principles are reflected design-wise in the use of open, standardised interfaces and ontologies. The principles are relevant in application development and deployment when a combination of distributed multiple services can provide value to a user in completing a use case satisfactorily.

1.2.2 Flexibility

As there exist a significant variety of user scenarios, requirements and use cases in predictive toxicology, flexibility is a key principle incorporated into OpenTox. Through the use of a component-based approach and the incorporation of the interoperability principles, many different and customised applications can be assembled that are based on the underlying platform.

1.2.3 Transparency

To achieve the scientific objective of knowledge-based enquiry based on principles of reasoning, reproducibility, and reliability, OpenTox supports the principle of Transparency in its design. Computational models should be available for scrutiny by other scientists in as complete a manner and detail as possible. Evaluators and regulators should be able to both understand the details and accurately reproduce the results of predictive toxicity

⁸ ambit.sourceforge.net/qmrf/jws/qmrfeditor.jnlp

models, and be able to reliably form judgements on their validity as evidence. The principle also supports achievement of the OECD validation principles such as an unambiguous algorithm and a mechanistic interpretation, if possible. Use of Open Source, Open Interfaces and Standards within OpenTox support implementation of this Transparency principle applied to *in silico*-based predictive toxicology applications and their reported results.

1.2.4 Extensibility

The field of predictive toxicology is rapidly developing and broadening in many areas including the use of biomarkers, systems biology, epigenetics, toxicokinetics, *in vitro* assays, stem cell technology, computational biology etc. Hence, OpenTox needs to be extensible to a broad range of future predictive toxicology applications. In such applications, contributing and diverse experimental data and models need to be combined as evidence supporting integrated testing, safety assessment and regulatory reporting as stipulated under REACH. In the initial design of the OpenTox Framework we have attempted to design a general solution for (Q)SAR model development and application. We also will address and strengthen its extensibility in subsequent project activities, and guided by suitable use cases, to additional areas of scientific enquiry in the predictive toxicology field as part of its evolutionary development.

2. Implementation Principles

OpenTox is an open source project and tries to follow the best practices of open source project management. This means that source code, technical discussions and documents are open to the general public and interested parties can participate in development. Within the design of the framework prototype we have concentrated on a number of principles:

- Open Source
- Open and Distributed Access
- Open Interfaces

In the following sections we describe these implementation principles in more detail.

2.1 Open Source

As the open source philosophy is inherently important for this project, all tools developed are openly available via public repositories. For example, details on current test services can be found on the OpenTox development testing web pages⁹. Within the framework, a variety of programming languages have been employed, such as Java, Ruby, and C++.

2.2 Open and Distributed Access

All current OpenTox web services adhere to the REpresentational State Transfer (REST) web service architecture¹⁰ for sharing data and functionality among loosely-coupled, heterogeneous systems. The REST architecture is based on five key principles:

1. Every resource can be uniquely identified;
2. Use standard HTTP;
3. Allow multiple representations of resources;
4. Use hypertext links for linking of resources;
5. Communicate statelessly.

Adhering to these principles, the REST web service architecture has a number of desired advantages when compared to other web service architectures:

1. It is lightweight, as only some additional xml mark-up is required;
2. The produced results are human readable, i.e. the resources are uniquely identified by URIs and described by representations;
3. RESTful web services are typically stateless and scalable;
4. The produced web services have a uniform interface (the only allowed operations are the HTTP operations);
5. Components manipulate resources by exchanging representations of the resources.

The choice of employing web services allows the complete framework to operate in different locations, independent of operating systems and underlying implementation details.

⁹ www.opentox.org/dev/testing/testtoxservices

¹⁰ Fielding, R.T., *Architectural Styles and the Design of Network-based Software Architectures*, Ph.D. dissertation, in University of California, Irvine. 2000

2.3 Open Interfaces

The publicly available OpenTox application programming interface (API) allows the cheminformatics and bioinformatics communities to participate in the development of new algorithms. It furthermore allows the independent comparisons of algorithms and models. We describe the current OpenTox API 1.1 in the next section, with respect to design issues and interoperability.

3. OpenTox API

To assure reliable interoperability between the various web services, a well-defined API is required. The API specifies how each OpenTox web service can be used, and how the returned resources look like. It further specifies the HTML status codes returned in case of succeeded operations as well as errors codes.

This section describes the OpenTox API version 1.1, the second version of the OpenTox API that was completed and published on the OpenTox website in November 2009. A short overview is given below, as well as a listing of all components including REST operations.

3.1 Overview

Figure 3.1 shows the OpenTox resources modelled in the OpenTox Ontology. These resources are provided by the various OpenTox web services. The links between the components reflects interaction between the respective web services.

The model web service (3.2.5) provides access to (prediction) models. Models are created via the algorithm web service (3.2.4), that supports different types of algorithms (e.g. supervised learning, feature selection, descriptor calculation, and data cleanup). Building a model will normally require various parameters, one/several dataset/s, as well as a set of features.

Datasets are stored in the dataset web service (3.2.3). A dataset contains data entries, which are chemical compounds, as well as their feature values. Features are defined as an object representing a property of a compound, including descriptors and calculated features, endpoints, and predictions. Different representations for chemical compounds can be accessed from the compound web service (3.2.1). The feature web service (3.2.2) provides the available features (e.g. structural features, chemical descriptors, endpoints).

The validation web service (3.2.6) evaluates and compares the performance of prediction models. Simple training-test-set-validation is supported as well as cross-validation. The validation result contains quality figures as defined in 1.1.4. The service further provides reports (available in html, PDF ...) that visualize the validation results.

The task web service (3.2.7) supports long-running, asynchronous processes. The ontology web service (3.2.8) provides meta information from relevant ontologies (which can be accessed using SPARQL queries¹¹), as well as lists of available services. Approaches to Authentication and Authorization will be specified in the next version of the API.

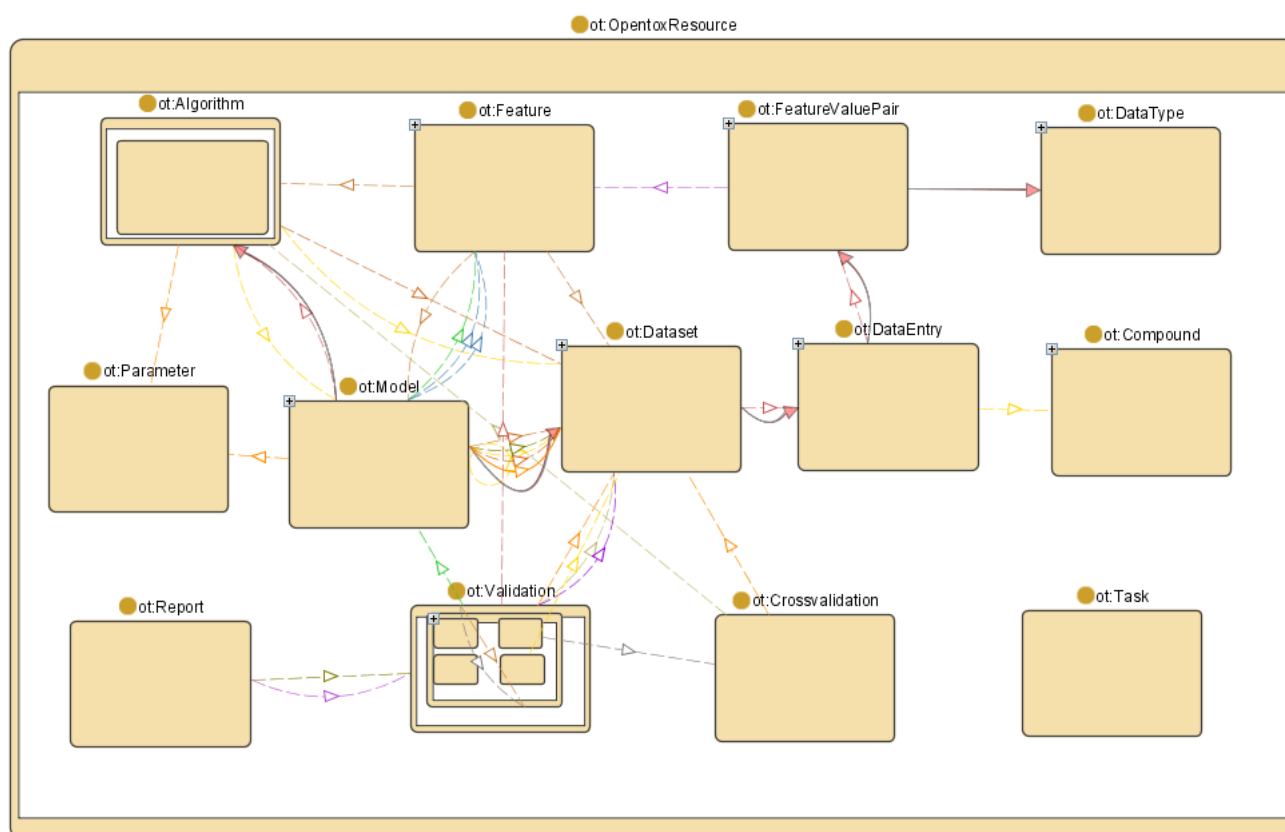


Figure 3.1 Relationships between OpenTox resources, modeled in OpenTox ontology.

3.2 Web services

This section describes REST operations and HTML status codes for OpenTox web services. Additional information (e.g. data representation format) can be found on the OpenTox API webpage¹².

¹¹ www.w3.org/TR/rdf-sparql-query/

¹² www.opentox.org/dev/apis/api-1.1

3.2.1 Compound

The Compound API provides different representations for chemical compounds with a unique and defined chemical structure.

REST operations

Description	Method	URI	Parameters	Result	Status codes
Search for compounds	GET	/compound	?search=value&sameas=URI_FROM_AN_ONTOLOGY	List of compounds, matching the query.	200,404,503
Get the representation of a compound	GET	/compound/{id}	(optional) ?feature_uris[]="URI to features"	Compound representation in one of the supported MIME formats; if feature_uris[] provided includes features and values.	200,404,503
Create a new compound	POST	/compound	Compound representation in a supported MIME format	URIs for new compounds.	200,400,503
Update a compound (optional)	PUT	/compound/{id}	Compound representation in a supported MIME format	-	200,400,404,503
Delete a compound (optional)	DELETE	/compound/{id}	-	- Delete all feature values per compound recursively?	200,400,404,503

Features per Compound

Description	Method	URI	Parameter	Result	Status codes
Get available feature URIs for a compound	GET	/compound/{cid}/feature	?feature_uris[]="URI to features" (optional)	Returns representation of the features as uri-list or RDF All available features are returned, if no parameter is specified.	200,404,503
Create a new feature value	POST	/compound/{cid}/feature	?feature_uri="URI to feature" (mandatory, single feature)&value=the_value	URI of the compound with the new feature, e.g. /compound/{id}?feature_uris[]=the-new-feature	200,400,503
Update a new feature value	PUT	/compound/{cid}/feature	?feature_uri="URI to feature" (mandatory, single feature)&value=the_value		200,400,404,503
Delete specified features from the compound	DELETE	/compound/{cid}/feature	?feature_uris[]="URI to features" (optional)		200,400,404,503

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
Compound not found	404	Not Found
Incorrect MIME type	400	Bad request
Service not available	503	Service unavailable

3.2.1.1 Conformers (optional)

The Conformers API provides [Optional] support for multiple (e.g. 3D) structures per chemical compound (single structure by default).

REST operations

Description	Method	URI	Parameters	Result	Status codes
Get available structures of a compound	GET	/compound/{id1}/conformer/	–	List of structure URIs.	200,404,503
Create a new structure	POST	/compound/{id1}/conformer	Representation in a supported MIME format.	New URI /compound/{id1}/conformer/{id2}	200,400,503
Remove all structures	DELETE	/compound/{id1}/conformer/	–	–	200,400,404,503
Get the representation of a structure	GET	/compound/{id1}/conformer/{id2}	?feature_uris[]="URI to features"	Representation in a supported MIME format, with feature values, if feature_uris[] provided.	200,404,503
Update the representation of a structure	PUT	/compound/{id1}/conformer/{id2}	Representation in a supported MIME format.	URI /compound/{id1}/conformer/{id2}	200,400,404,503
Remove a structure	DELETE	/compound/{id1}/conformer/{id2}	–	–	200,400,404,503

Features per Conformer

Description	Method	URI	Parameter	Result	Status codes
Get available feature URIs for a compound	GET	/compound/{cid}/conformer/{cid}/feature	?feature_uris[]="URI to features" (optional)	Returns representation of the features as uri-list or RDF. All available features are returned, if no parameter is specified.	200,404,503
Create a new feature value	POST	/compound/{cid}/conformer/{cid}/feature	?feature_uri="URI to feature" (mandatory, single feature)&value=the_value	URI of the compound with the new feature, e.g. /compound/{id}/conformer/{cid}?feature_uris[]=the-new-feature	200,400,503

Update a new feature value	PUT	/compound/{cid}/conformer/{cid}/feature	?feature_uri="URI to feature" (mandatory, single feature)&value=the_value		200,400,404,503
Delete specified features from the compound	DELETE	/compound/{cid}/conformer/{cid}/feature	?feature_uris[]="URI to features" (optional)		200,400,404,503

HTTP status codes

See HTTP status codes for compounds.

3.2.2 Feature

A Feature is an object, representing any kind of property, assigned to a Compound. The feature types are determined via their links to ontologies (Feature ontologies, Descriptor ontologies, Endpoints ontologies).

REST operations

Description	Method	URI	Parameters	Result	Status codes
get description of a specific feature definition	GET	/feature/{id}	–	URI-list or RDF representation of a feature.	200,404,503
create a new feature	POST	/feature	Content-type ="any-of-RDF-types", content=RDF-representation	URI of the new feature definition.	200,400,404,503
update feature	PUT	/feature/{id}	Content-type ="any-of-RDF-types", content=RDF-representation	–	200,400,404,503
delete feature	DELETE	/feature/{id}	–	–	200,400,404,503
get a list of available feature definitions	GET	/feature	?query=URI-of-the-owl:sameAs-entry	URI list or RDF of features found by the query or all available, if query is empty. Returns all features, for which owl:sameAs is given by the query.	200,404,503

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
No feature found, or specific feature not found	404	Not Found
Incorrect parameters	400	Bad request
Service not available	503	service unavailable

3.2.3 Dataset

The Dataset API provides access to chemical compounds and their features (e.g. structural, physical–chemical, biological, and toxicological properties)

REST operations

Description	Method	URI	Parameters	Result	Status codes
Get a list of available datasets	GET	/dataset	Query parameters (optional, to be defined by service providers).	List of URIs or RDF for the metadata only.	200,404,503
Get a dataset	GET	/dataset/{id}	–	Representation of the dataset in a supported MIME type.	200,404,503
Query a dataset	GET	/dataset/{id}	compound_uris[] and/or feature_uris[] to select compounds and features; further query parameters may be defined by service providers.	Representation of the query result in a supported MIME type.	200,404,503
Get metadata for a dataset	GET	/dataset/{id}/metadata	–	Representation of the dataset metadata in a supported MIME type.	200,404,503
Get a list of all compounds in a dataset	GET	/dataset/{id}/compounds	–	List of compound URIs.	200,404,503
Get a list of all features in a dataset	GET	/dataset/{id}/features	–	RDF or List of feature URIs (pointing to feature definitions/ontologies).	200,404,503
Create a new dataset	POST	/dataset	Dataset representation in a supported MIME type. MIME type to be specified via: Content-type header. <ul style="list-style-type: none"> • <i>Content-type:application/www-form-urlencoded</i> dataset_uri, feature_uris[] and compound_uris[] parameters are used to specify subset of a dataset, as in GET operation; • File upload via <i>Content-type:multipart/form-data</i>: file parameter; • File upload metadata: parameters as in opentox.owl 	New URI /dataset/{id} or redirect to task URI (for large uploads).	200,202,400,503

Update a dataset	PUT	<ul style="list-style-type: none"> • /dataset/{id} 	<ul style="list-style-type: none"> • Data representation in a supported MIME type; • <i>entries for existing compound/feature pairs will be overwritten, entries for new compound/features will be added;</i> • File upload metadata: Dublin core annotation parameters, as in opentox.owl#Dataset • <i>Content-type:application/www-form-urlencoded</i> dataset_uri , feature_uris[] and compound_uris[] parameters are used to specify subset of a dataset, as in GET operation; • File upload via <i>Content-type:multipart/form-data</i>: file parameter • File upload metadata: Dublin core annotation parameters, as in opentox.owl#Dataset 	Dataset URI or task URI.	200,202,400,404,503
Remove a dataset	DELETE	/dataset/{id}	–	–	200,404,503
Remove a part of the dataset	DELETE	/dataset/{id}	compound_uris[] and/or feature_uris[] ; further query parameters may be defined to select the data to be deleted.	–	200,404,503

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
Asynchronous task started	202	Accepted
Dataset not found	404	Not Found
Incorrect MIME type	400	Bad request
Service not available	503	Service unavailable

3.2.4 Algorithm

The Algorithms API Provides access to OpenTox algorithms.

REST operations

Description	Method	URI	Parameters	Result	Status codes
Get URIs of all available algorithms	GET	/algorithm	(optional) ?sameas=URI-of-the-owl:sameAs-entry	List of all algorithm URIs or RDF representation, or algorithms of specific types, if query parameter exists. Returns all algorithms, for which owl:sameAs is given by the query.	200,404,503
Get the ontology representation of an algorithm	GET	/algorithm/{id}	–	Algorithm representation in one of the supported MIME types.	200,404,503
Apply the algorithm	POST	/algorithm/{id}	dataset_uri parameter prediction_feature , more to be specified and documented by algorithm provider dataset_service =datasetseviceuri	<i>model URI</i> <i>dataset URI</i> <i>featureURI</i> Redirect to task URI for time consuming computations.	200,303,404,503

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
No algorithm in the respective category found, or specific algorithm not found	404	Not Found
Incorrect dataset URI, or incorrect parameters	400	Bad request
Model building error	500	Internal Server Error
Model building in progress (redirect to task URI)	303	Redirect
Service not available	503	Service unavailable

3.2.5 Model

The Model API provides access to OpenTox prediction models.

REST operations

Description	Method	URI	Parameters	Result	Status codes
Get a list of all available models	GET	/model	(optional) ?query=URI-of-the-owl:sameAs-entry	List of model URIs or RDF representation. If query specified, returns all	200,404,503

				models, for which owl:sameAs is given by the query.	
Get the representation of a model	GET	/model/{id}	–	Representation of the model in a supported MIME type.	200,404,503
Delete a model	DELETE	/model/{id}	–	–	200,404,503
Apply a model to predict a dataset	POST	/model/{id}	dataset_uri result_dataset =dataseturi dataset_service =datasetseviceuri	URI of created prediction dataset (predictions are features), task URI for time consuming computations.	200,202,400,404,500,503
Apply a model to predict a compound	POST	/model/{id}	compound_uri	Prediction in a supported MIME type; task URI for time consuming computations.	200,202,400,404,500,503

Model variables

REST operations

Description	Method	URI	Parameters	Result	Status codes
List of independent variables	GET	/model/{id}/independent	–	URI-list/RDF of features used as independent variables.	200,404,503
List of dependent variables	GET	/model/{id}/dependent	–	URI-list/RDF of features used as dependent variables.	200,404,503
List of predicted features	GET	/model/{id}/predicted	–	URI-list/RDF of features, where predictions are stored.	200,404,503

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
Asynchronous task accepted	202	Accepted
Dataset_id is wrong	400	Bad Request
Model for specific id not found	404	Not Found
Prediction error	500	Internal server error
Service not available	503	Service unavailable

3.2.6 Validation

3.2.6.1 Standard Validation

A validation corresponds to the validation of a model on a test dataset. The results are stored in another dataset. Parameters with default values are optional.

REST operations

Description	Method	URI	Parameters	Result	Status codes
Get all validations	GET	/	–	List of validation URIs.	200,404
Retrieves a validation representation	GET	/id}	–	Validation representation in one of the supported MIME types.	200,404
Validates a model on a test dataset	POST	/	model_uri test_dataset_uri	Validation URI or Task URI.	200,400,404,500
Builds a model on a training dataset and validates it on a test dataset	POST	/	algorithm_uri prediction_feature algorithm_params (string, default="") training_dataset_uri test_dataset_uri y_scramble (boolean, default=false) y_scramble_seed (integer, default=1)	Validation URI or Task URI.	200,400,404,500
Splits a dataset into training and test dataset according to a certain ratio, and performs a validation	POST	/training_test_split	algorithm_uri prediction_feature algorithm_params (string, default="") dataset_uri split_ratio(float, default=0.66) random_seed(integer, default=1) y_scramble (boolean, default=false) y_scramble_seed (integer, default=1)	Validation URI or Task URI.	200,400,404,500
<i>OPTIONAL:</i> Performs a bootstrap validation	POST	/bootstrap	algorithm_uri prediction_feature dataset_params (string, default="") dataset_uri bootstrap_percentage(float, default=0.66) random_seed(integer, default=1) y_scramble (boolean, default=false) y_scramble_seed (integer, default=1)	Validation URI or Task URI.	200,400,404,500
Deletes a validation.	DELETE	/id}	–	–	200,404

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
Validation not found	404	Not Found
Illegal model/algorithm/dataset/algorithm params	400	Bad request
Validation/prediction error	500	Internal Server Error

3.2.6.2 Cross-Validation

Performs a k -fold cross-validation, resulting in k validation resources. Parameters with default values are optional.

REST operations

Description	Method	URI	Parameters	Result	Status codes
Get all cross-validations	GET	/crossvalidation	–	List of cross-validation URIs.	200,404
Retrieves a cross-validation representation	GET	/crossvalidation/{id}	–	Cross-Validation in one of the supported MIME types.	200,404
Returns all (k) validations that belong to a crossvalidation	GET	/crossvalidation/{id}/validations	–	List of validation URIs.	200,404
Performs a k-fold cross-validation	POST	/crossvalidation	algorithm_uri prediction_feature algorithm_params (string, default="") num_folds (integer, default=10) random_seed (integer, default=1) stratified (boolean, default=true) y_scramble (boolean, default=false) y_scramble_seed (integer, default=1)	Cross-Validation URI or Task URI.	200,400,404,500
Performs a leave-one-out cross-validation	POST	/crossvalidation/loo	algorithm_uri prediction_feature algorithm_params (string, default="") y_scramble (boolean, default=false) y_scramble_seed (integer, default=1)	Cross-Validation URI or Task URI.	200,400,404,500
Deletes a cross-validation	DELETE	/crossvalidation/{id}	–	–	200,404

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
Cross validation not found	404	Not Found
Illegal model/algorithm/dataset/algorithm params	400	Bad request
Validation/prediction error	500	Internal Server Error

3.2.6.3 Validation – Report

The validation report visualizes the (prediction) results of algorithms.

REST operations

Description	Method	URI	Parameters	Result	Status codes
Get all report types	GET	/report	–	List of available report types.	200,404
Get all reports for the particular report type	GET	/report/{report-type}	–	List of available reports as URI.	200,404
Retrieves a report in XML / PDF / HTML / RTF format	GET	/report/{report-type}/{id}	–	Report in specified format.	200,404
Creates a report	POST	/report/{report-type}	<i>various params, see below</i>	Report URI or Task URI.	200,400,404,500
Deletes a report	DELETE	/report/{report-type}/{id}	–	–	200,404
Available (validation-)report types					
Create ToxPredict report (multiple models, one compound to predict)	POST	/report/toxpredict	List of validation URIs.	Report URI or Task URI	200,400,404,500
Create single validation report (one model, one test dataset)	POST	/report/validation	Validation URIs.	Report URI or Task URI	200,400,404,500
Create cross-validation report (crossvalidation with one algorithm and one dataset)	POST	/report/crossvalidation	Cross-validation URIs.	Report URI or Task URI	200,400,404,500
Create report for comparing different prediction algorithms (cross-validations/validations with multiple algorithms and datasets)	POST	/report/algorithm_comparison	List of cross-validation URIs or list of validation URIs.	Report URI or Task URI	200,400,404,500
Create report for comparing different models	POST	/report/model_comparison	List of validation URIs.	Report URI or Task URI	200,400,404,500
Special report formats					
Create QMRF report	POST	/report/qmrf	Model URI or a List of cross-validation URIs and/or validation URIs of the same model; additional fields of the report that cannot be filled out automatically (yet to be defined).	Report URI or Task URI	200,400,404,500
Create QPRF report	POST	/report/qprf	Model URI or a List of cross-validation URIs and/or validation URIs of	Report URI or Task URI	200,400,404,500

			the same model; additional fields of the report that cannot be filled out automatically (yet to be defined).		
--	--	--	--	--	--

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
Report type / report not found	404	Not Found
Illegal params	400	Bad request
Error creating the report	500	Internal Server Error

3.2.7 Task

Asynchronous jobs are handled via an intermediate Task resource. A resource, submitting an asynchronous job should return the URI of the task.

REST operations

Description	Method	URI	Parameters	Result	Status codes
Get a list of all available tasks	GET	/task	?query=task status as in opentox.owl	List of URIs/RDF representation.	200,503,401
Get the representation of a running task	GET	/task/{id}	-	Task representation in one of the supported MIME formats.	202,404,503,401
Get the representation of a completed task	GET	/task/{id}		The URI of the newly created resource in the Location header.	303,404,503
Delete a task	DELETE	/task/{id}			200, 404, 503,401

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
Task resource created	201	The request has been fulfilled and resulted in a new resource being created.
The Task is not completed	202	accepted, processing has not been completed
The Task is completed	303	The task is completed, the URL where the resource is available is in the Location header (mandatory)
task_id is wrong	400	Bad Request
Not Authorized	401	Not Authorized
Task for specific id not found	404	Not Found
Error	500	Internal server error
Service not available	503	Service unavailable

3.2.8 Ontology Service

The Ontology Service provides storage and search functionality for objects, defined in OpenTox services and relevant ontologies

REST operations

Description	Method	URI	Parameters	Result	Status codes
Retrieve SPARQL query results	GET	/ontology	?query =SPARQL_QUERY (mandatory)	RDF representation of the query results.	200,404,500
Predefined query to retrieve all models	GET	/ontology/models		RDF representation of all models.	
Predefined query to retrieve all endpoints	GET	/ontology/endpoints		RDF representation of all endpoints.	
Predefined query to retrieve all algorithms	GET	/ontology/algorithms		RDF representation of all algorithms.	
Submit SPARQL query and/or OpenTox service URL	POST	/ontology	uri =URL of a OpenTox RDF resource query =SPARQL_QUERY	RDF representation of the query results, if query is specified. if uri is specified, the server retrieves a RDF representation and adds it to the RDF storage, thus making it available for the subsequent queries. Any non-empty subset of parameters is valid (i.e. only query, only model_uri, query and algorithm_uri, etc.).	200,404,500,502

HTTP status codes

Interpretation	Nr	Name
Success	200	OK
Wrong query syntax	404	Bad request
	500	Internal server error
Error when retrieving RDF representation from specified URL	502	

4. Prototype Use Cases

We identified two initial use cases for the implementation of the OpenTox Framework prototype. The first case, **ToxPredict**, is aimed at the user having no or little experience in QSAR predictions. This use case should offer an easy-to-use user interface, allowing the user to enter a chemical structure and to obtain in return a toxicity prediction for one or more endpoints. The second case, **ToxCreate**, is aimed at the experienced user, allowing them to construct and to validate models using a number of datasets and algorithms.

Both use cases also demonstrate inter-connectivity between multiple partner services. Within **ToxPredict**, several web services from partners TUM, IDEA, and NTUA are operating together, while in **ToxCreate** the model construction is performed using partner IST's web services, while the validation and reporting is executed using partner ALU's services. An important subsequent step to be pursued in forthcoming development iterations will be the interoperation of ToxCreate and ToxPredict across the combined services of five or more partners.

Within this section, we also provide a more technical use case of building and validating a model. The use case is one of the underlying use cases within the ToxPredict use case, where an algorithm trains a model on a training dataset, and then predicts the compounds of a test dataset with regards to a certain endpoint.

4.1 ToxPredict Use Case

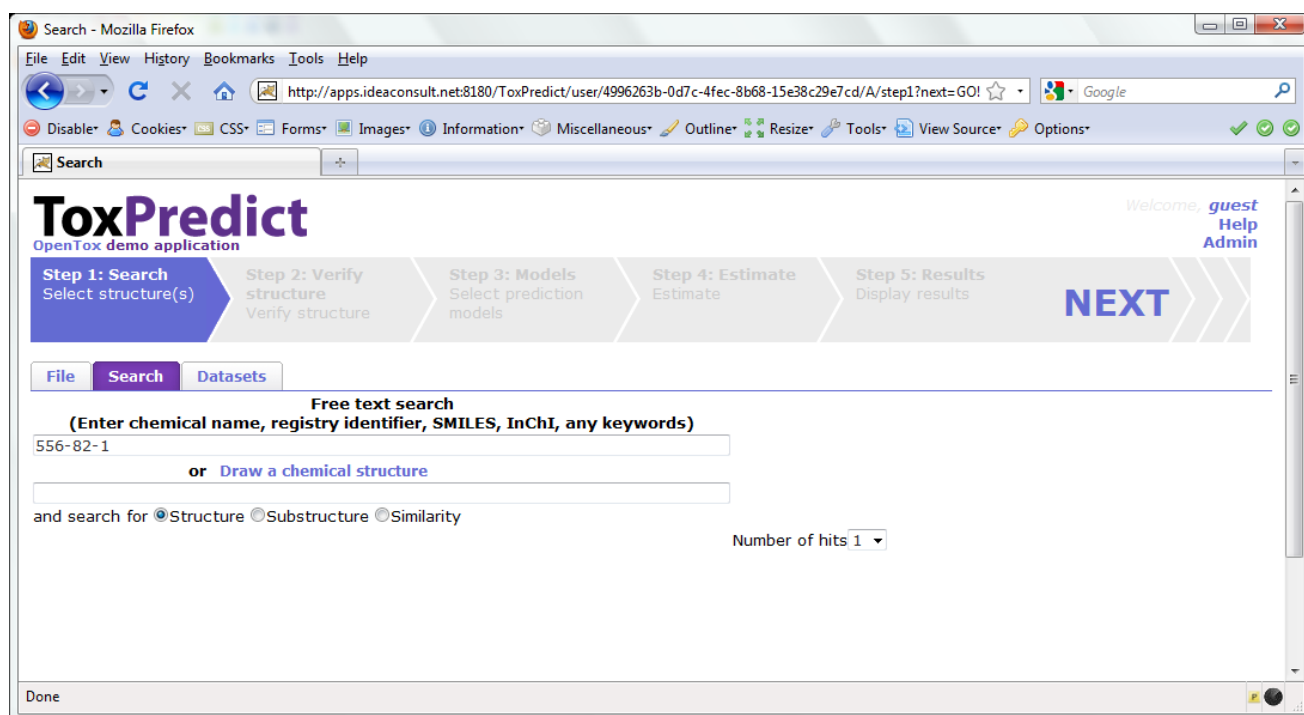
As the **ToxPredict** use case should offer easy access to estimate the toxicological hazard of a chemical structure for non-QSAR specialists, the main aim was to design a simple yet easy-to-use user interface. For this, one of the aims was also to reduce the number of possible parameters the user has to enter when querying the service. The use case can be divided into the following five steps:

1. Enter/select a chemical compound
2. Display selected/found structures
3. Select models
4. Perform the estimation

5. Display the results

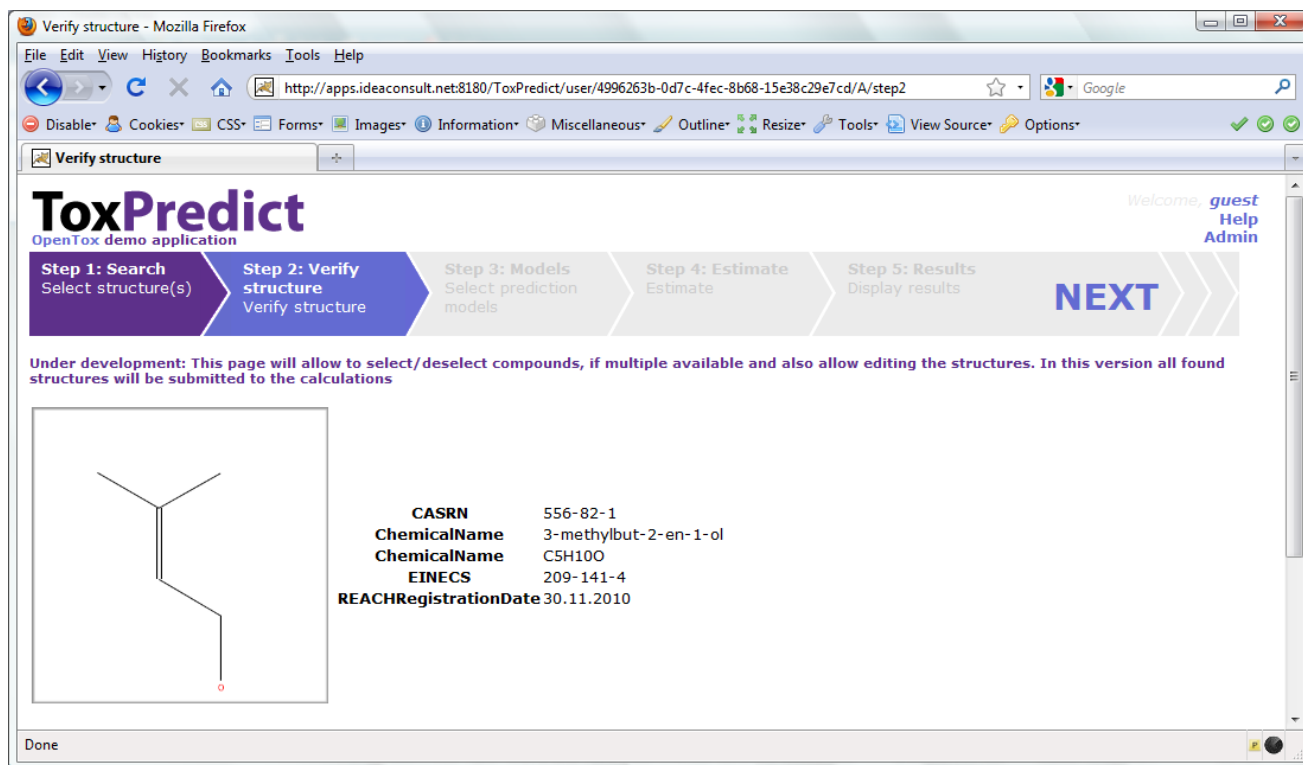
The following sequence of screenshots and descriptions explain the workflow and operations of a sample ToxPredict user session.

1. Enter/select a chemical compound



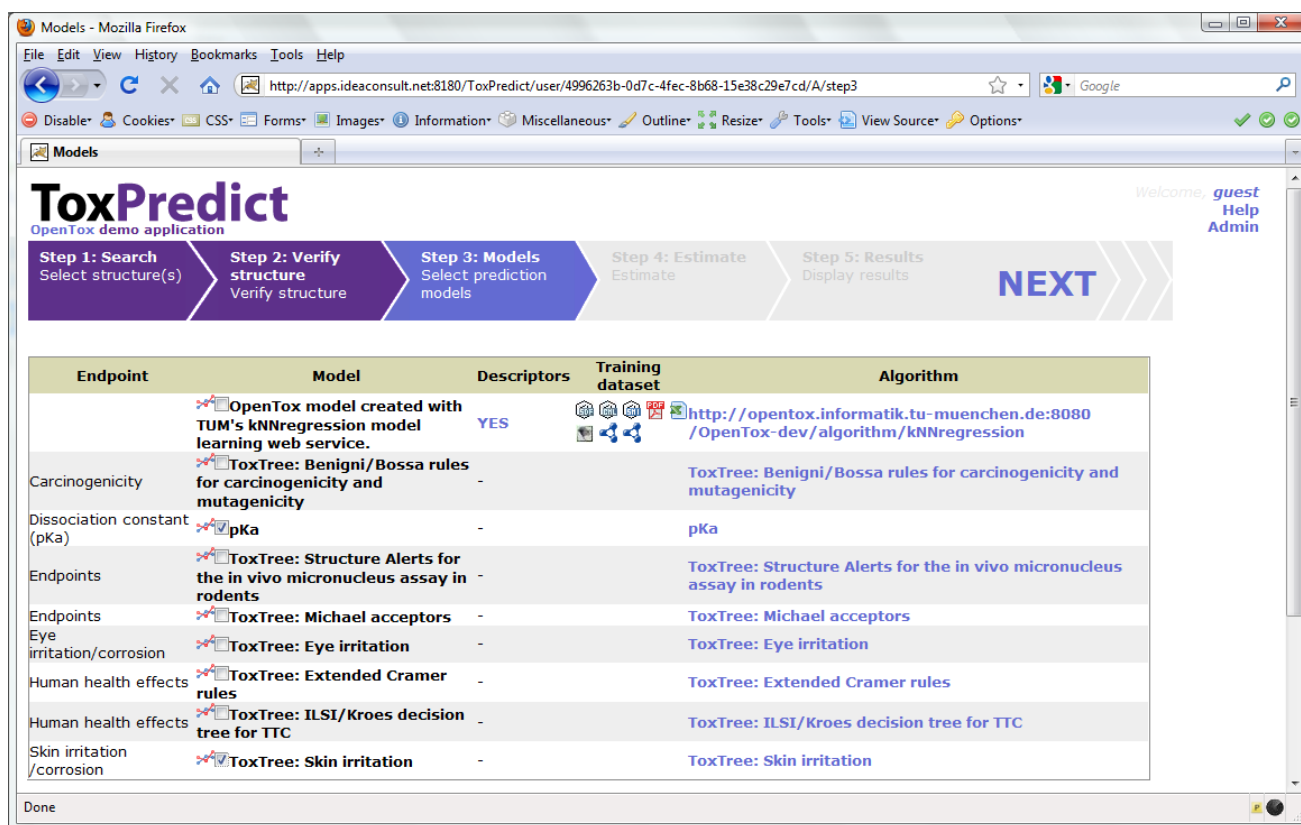
The first step in the ToxPredict workflow provides the means to specify the chemical structure(s) for further estimation of toxicological properties. Free text searching allows the user to find chemical compounds by chemical names and identifiers, SMILES and InChI strings, and any keywords available in the OpenTox database. The database contains information from multiple sources, including the ECHA pre-registration list, and was created within OpenTox WP3. Its content and procedures for curation are extensively described in the OpenTox D3.2 deliverable report (28 February 2010).

2. Display selected/found structures



The second step displays the chemical compounds, selected by the previous step. In the next release, this step will be updated to allow the selection/de-selection of structures, and editing of the structures and associated relevant information. The OpenTox REST Dataset services are used in this step of the application in order to retrieve the requested information.

3. Select models



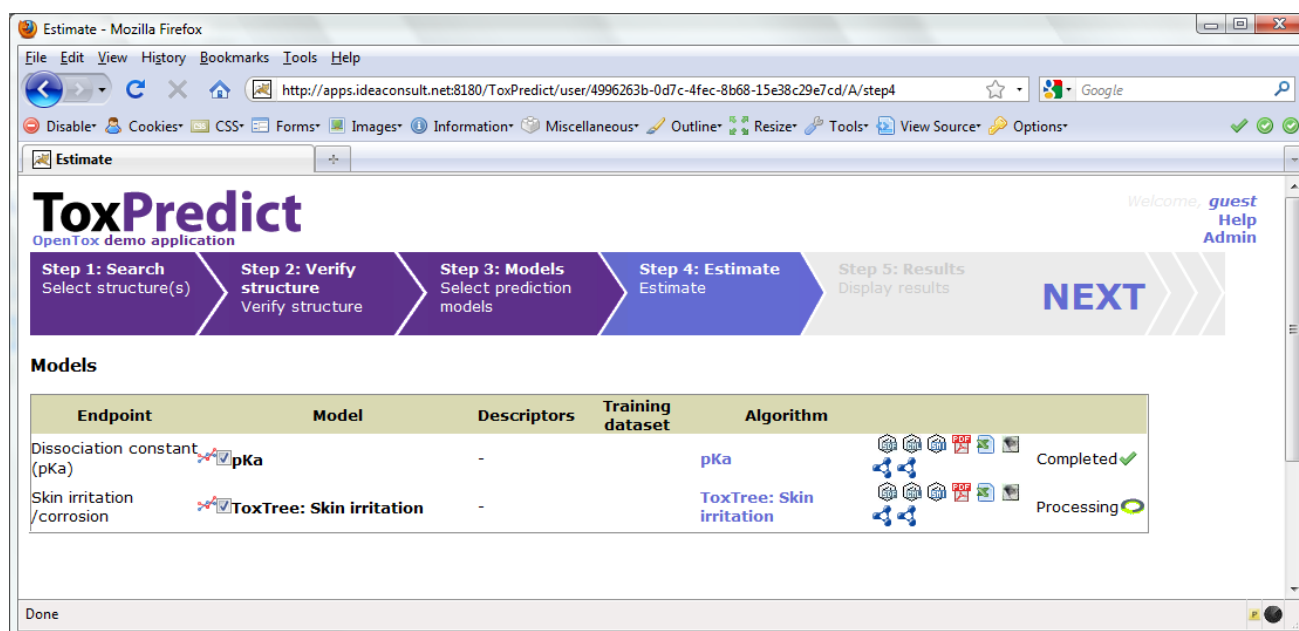
Endpoint	Model	Descriptors	Training dataset	Algorithm
	OpenTox model created with TUM's kNNregression model learning web service.	YES	http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/kNNregression	
Carcinogenicity	ToxTree: Benigni/Bossa rules for carcinogenicity and mutagenicity	-		ToxTree: Benigni/Bossa rules for carcinogenicity and mutagenicity
Dissociation constant (pKa)	pKa	-		pKa
Endpoints	ToxTree: Structure Alerts for the in vivo micronucleus assay in rodents	-		ToxTree: Structure Alerts for the in vivo micronucleus assay in rodents
Endpoints	ToxTree: Michael acceptors	-		ToxTree: Michael acceptors
Eye irritation/corrosion	ToxTree: Eye irritation	-		ToxTree: Eye irritation
Human health effects	ToxTree: Extended Cramer rules	-		ToxTree: Extended Cramer rules
Human health effects	ToxTree: ILSI/Kroes decision tree for TTC	-		ToxTree: ILSI/Kroes decision tree for TTC
Skin irritation /corrosion	ToxTree: Skin irritation	-		ToxTree: Skin irritation

In the third step, a list of available models is displayed. Links to training datasets, algorithms and descriptor calculation REST services are provided. The models provide information about the independent variables used, the target variables (experimental toxicity data) and predicted values. All these variables are accessible via the OpenTox Feature web service, where each feature can be associated with a specific entry from the existing endpoint ontology. The association is usually done during the upload of the training data into the database. The endpoint, associated with the model variables is automatically retrieved and displayed in the first column of the list. This provides an automatic and consistent way of complying with the first OECD validation principle of using a “Defined endpoint”.

This step involves an interplay between multiple OpenTox web services. Algorithm, Model, and Feature services are registered into the Ontology service, which provides RDF triple storage with SPARQL, allowing various queries. The ToxPredict application queries the Ontology service for all available models, along with the associated information about algorithms used in the model, descriptors, and endpoints. The list of models may include models, provided by different partners and running on several remote sites (TUM and IDEA models are shown in this example). The Ontology service serves like a hub for gathering a list of available models and algorithms from remote sites. There could be multiple

instances of the ToxPredict application, configured to use different Ontology services, and therefore, allowing for different subset of models to be exposed to end users.

4. Perform the estimation



Endpoint	Model	Descriptors	Training dataset	Algorithm	Status
Dissociation constant (pKa)	pKa	-	pKa		Completed ✓
Skin irritation /corrosion	ToxTree: Skin irritation	-	ToxTree: Skin irritation		Processing

Models, selected in Step3 are launched in Step 4, where the user can monitor the status of the processing. The processing status is retrieved via OpenTox Task services. Different Model, Algorithm, Dataset, and Ontology services, running on different remote locations can be involved at this stage. If a model relies on a set of descriptors, an automatic calculation procedure is performed, which involves launching a descriptor calculation by remote Algorithm services. The procedure is as follows:

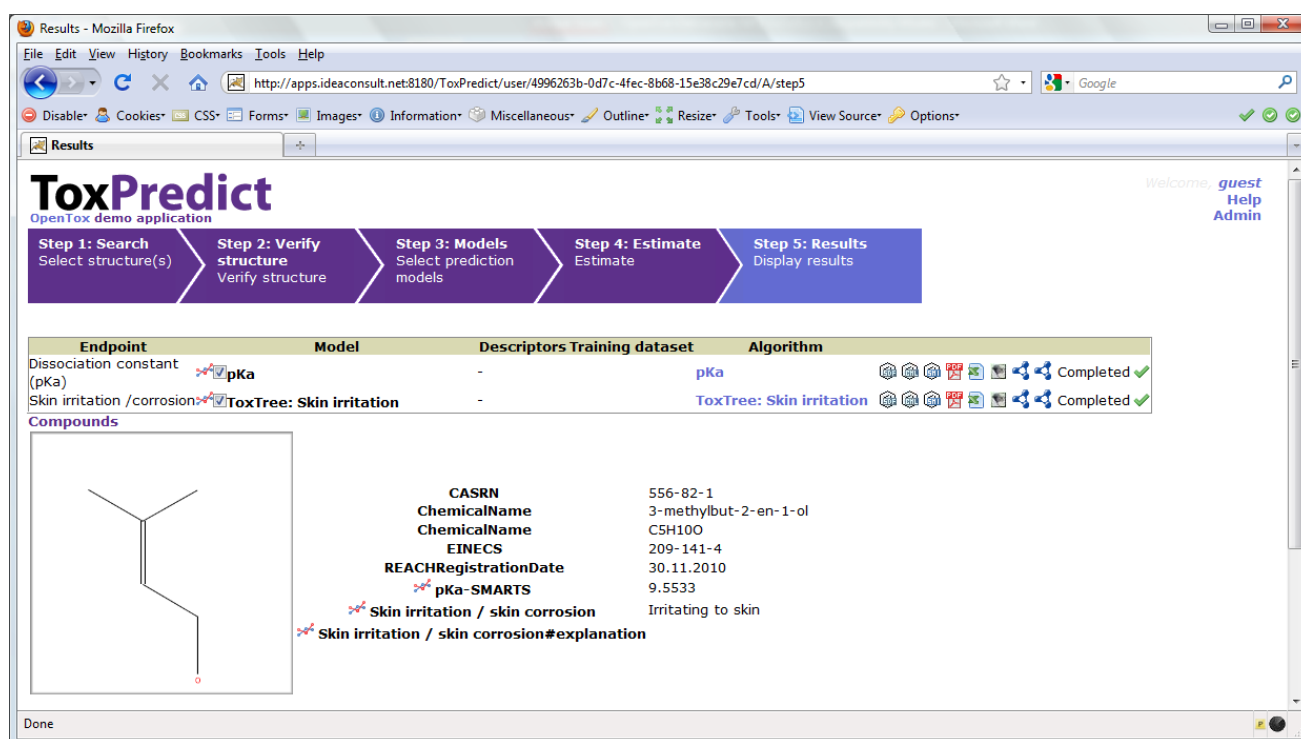
The Ontology service is queried to retrieve information about the independent variables, used in the model. If no such variables are involved (e.g., in case of ToxTree models, which rely on chemical structure only), the workflow proceeds towards model estimation. In case of a model, based on descriptors (e.g., a regression model), the procedure is slightly more complex, as explained below.

Each independent variable is represented as a Feature and managed via the Feature service. Each feature has associated a web address (OWL property `opentox:hasSource` from OpenTox OWL ontology), which specifies its origin. The tag could point to an OpenTox Algorithm or Model service, in case it holds a calculated value, or point to a Dataset service, in case it contains information, uploaded as a dataset (for example experimental endpoint data). If the feature origin is a descriptor calculation algorithm, the web address points to the Algorithm service, used to calculate descriptor values, and the same web

address can be used again via the OpenTox Algorithm API in order to calculate descriptors for user-specified structures. The Algorithm services perform the calculation and store results into a Dataset service, possibly at a remote location. Then finally, a dataset with all calculated descriptor values is submitted to the Model service. Upon estimation, Model results are submitted to a Dataset service, which could be at a remote location, which could be the same or different to that for the model services.

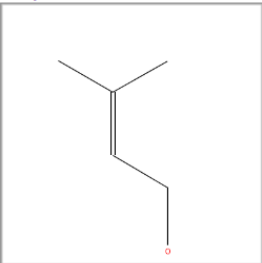
The interplay of multiple services, running on remote sites, provide a flexible means for the integration of models and descriptors, developed by different organisations and running in different environments. Identification of algorithms and models via web URLs ensure the compliance with the OECD validation principle 2 of “An unambiguous algorithm”, as well as repeatability of the results of the model building. Extensive meta information about the algorithm and models themselves is accessible via web URLs and the OpenTox API.

5. Display the results



Endpoint	Model	Descriptors	Training dataset	Algorithm
Dissociation constant (pKa)	pKa	-	pKa	Completed
Skin irritation / corrosion	ToxTree: Skin irritation	-	ToxTree: Skin irritation	Completed

Compounds



CASRN 556-82-1
ChemicalName 3-methylbut-2-en-1-ol
EINECS C5H10O
REACHRegistrationDate 209-141-4
pKa-SMARTS 30.11.2010
Skin irritation / skin corrosion 9.5533
Skin irritation / skin corrosion#explanation Irritating to skin

The final step displays estimation results, as well as compound identification and other related data. Initial demonstration reports in several formats can be accessed via icons on the right hand side of the browser display.

Next Steps in Development

We summarise here the next steps in our future work on this use case:

General: Improving the user interface, based on feedback from internal and external testers. The workflow design is generally considered very intuitive and convenient for users, although there are some concerns that too many steps may be involved.

Step 1: The user interface will be extended to provide the means for uploading files and reusing existing search results. The related service functionality is already available via the Dataset service.

Step 2: Plans include providing data retrieval from several third-party sources like IUCLID5 and PubChem via the standardized OpenTox Dataset service API. Communication with IUCLID5 via web services will improve the utility of the ToxPredict application in a REACH context.

Step 3: Future work includes solving several technical issues, in order to introduce models from all OpenTox partners, as well as providing wrappers for third-party models, in order to make them available via the OpenTox API and visible for the ToxPredict application.

Step 4: Task services will be extended to allow for cancelling long running estimations, as well as to provide more detailed information about processing status.

Step 5: The web page will be extended to include information about the relevant experimental endpoint values, retrieved from the database. The Dataset service, providing such functionality, is already available and can be quickly integrated into the workflow. An important functionality, that is currently missing, is inclusion of model validation statistics, which depend on integration of the Validation service, developed in OpenTox WP5. The Reporting service developed in WP5 will provide the means for generating reports in REACH-compliant format (QMRF or CSA format).

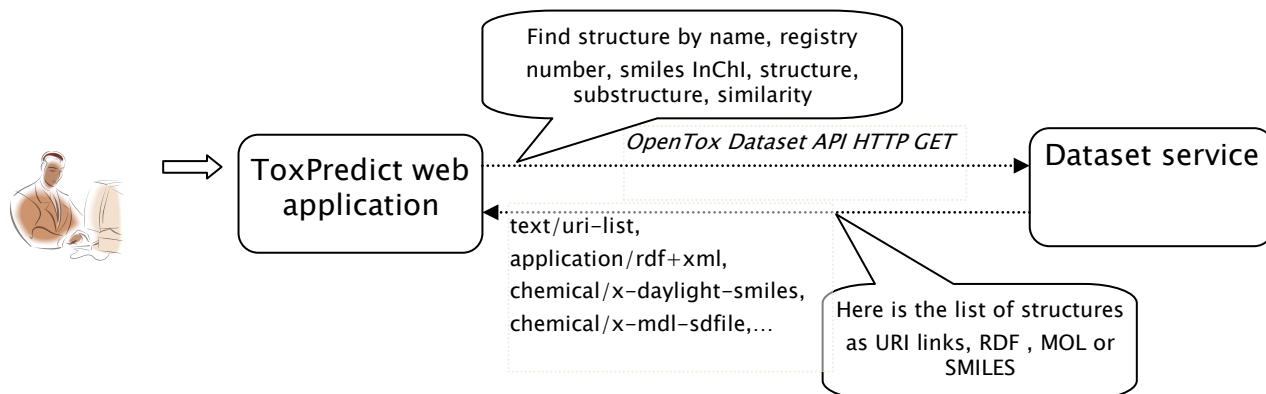
Conclusions

ToxPredict is a demonstration web application, providing a user-friendly interface for estimating toxicological hazards. It provides a front end to multiple OpenTox services, currently integrating IDEA ontology, dataset, feature and model services with TUM descriptor calculation and model services, and NTUA algorithm services. Future work will include integration of other partners and third party model services, as well as the reports, generated by the ALU-FR Validation and Reporting service. While current functionality may appear to an end-user not much different from a stand-alone prediction application like ToxTree, the back-end technology provides a very flexible means for integrating datasets, models and algorithms, developed by different software technologies and organisations and running at remote locations.

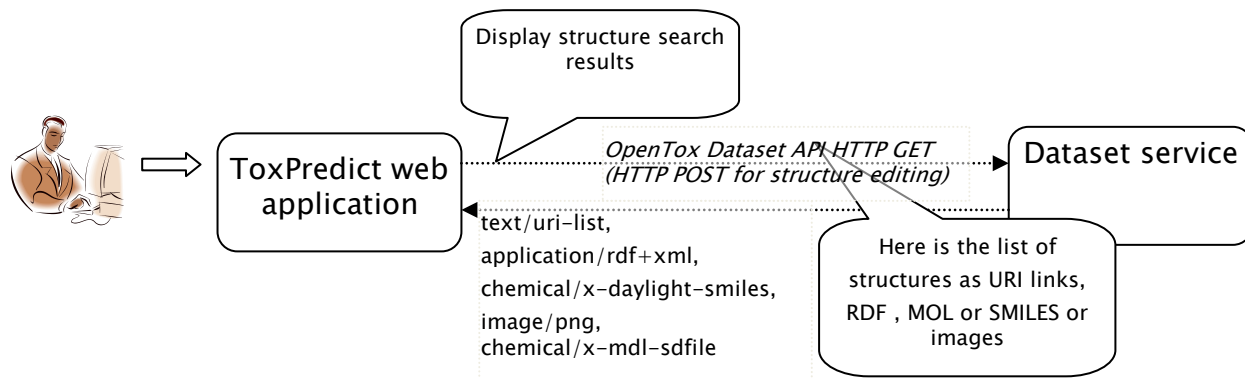
4.1.1 Interaction of OpenTox services, employed in ToxPredict

This section describes visually the interaction and sequence of OpenTox services interoperating during the different steps of the ToxPredict application execution.

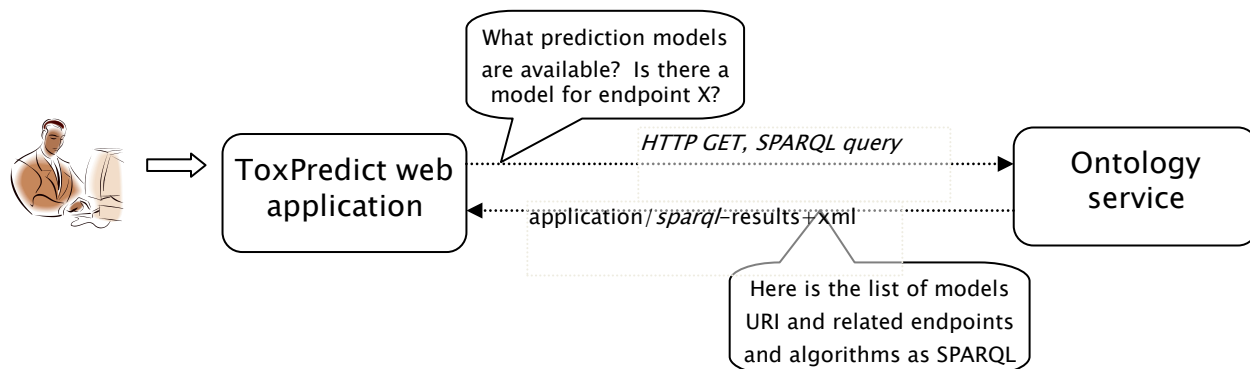
Step 1 – Enter Compound



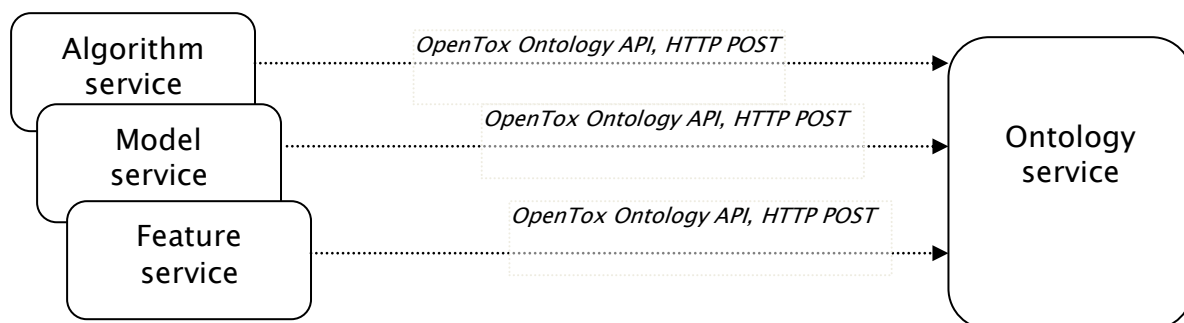
Step 2 – Structure selection



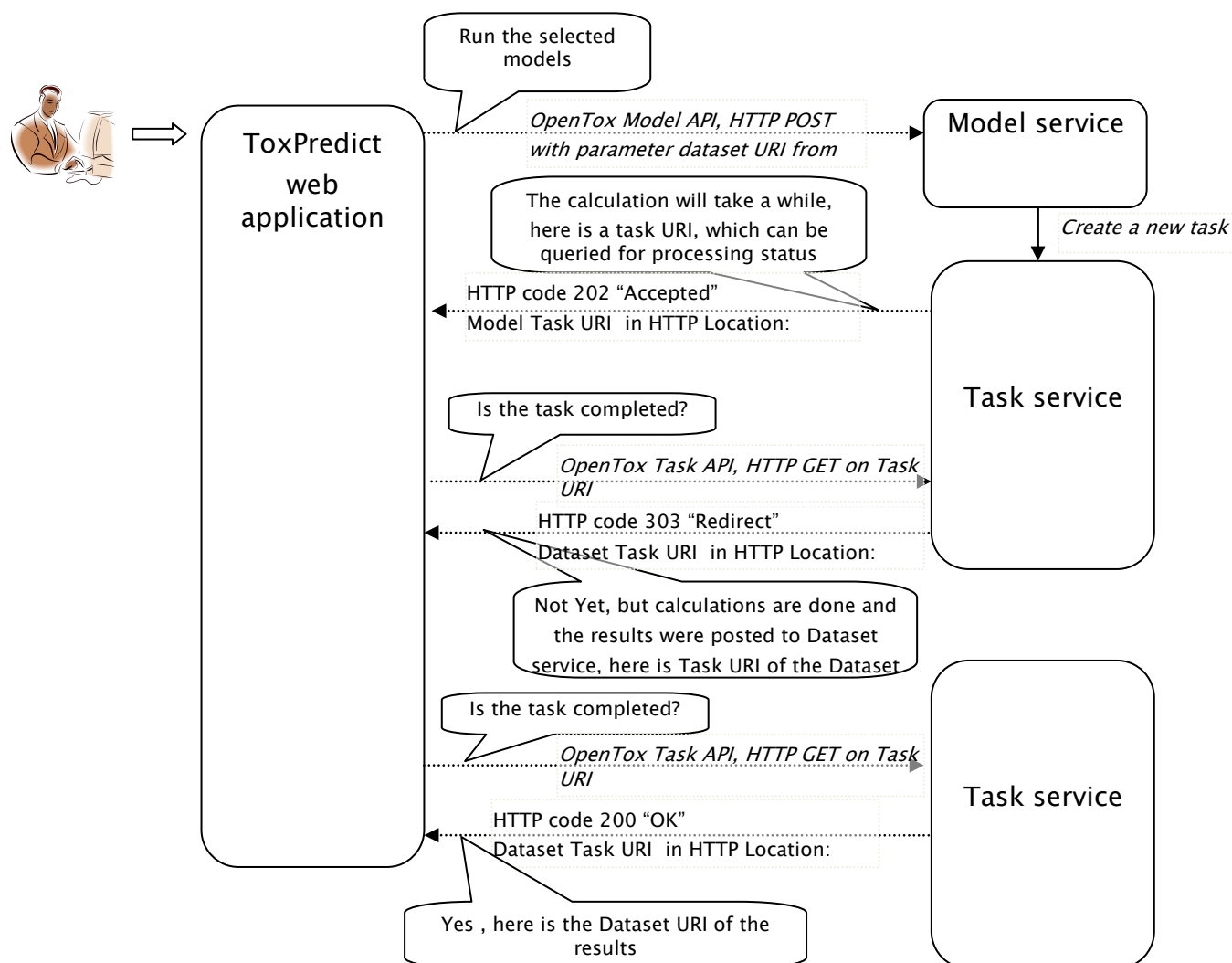
Step 3 – Model selection



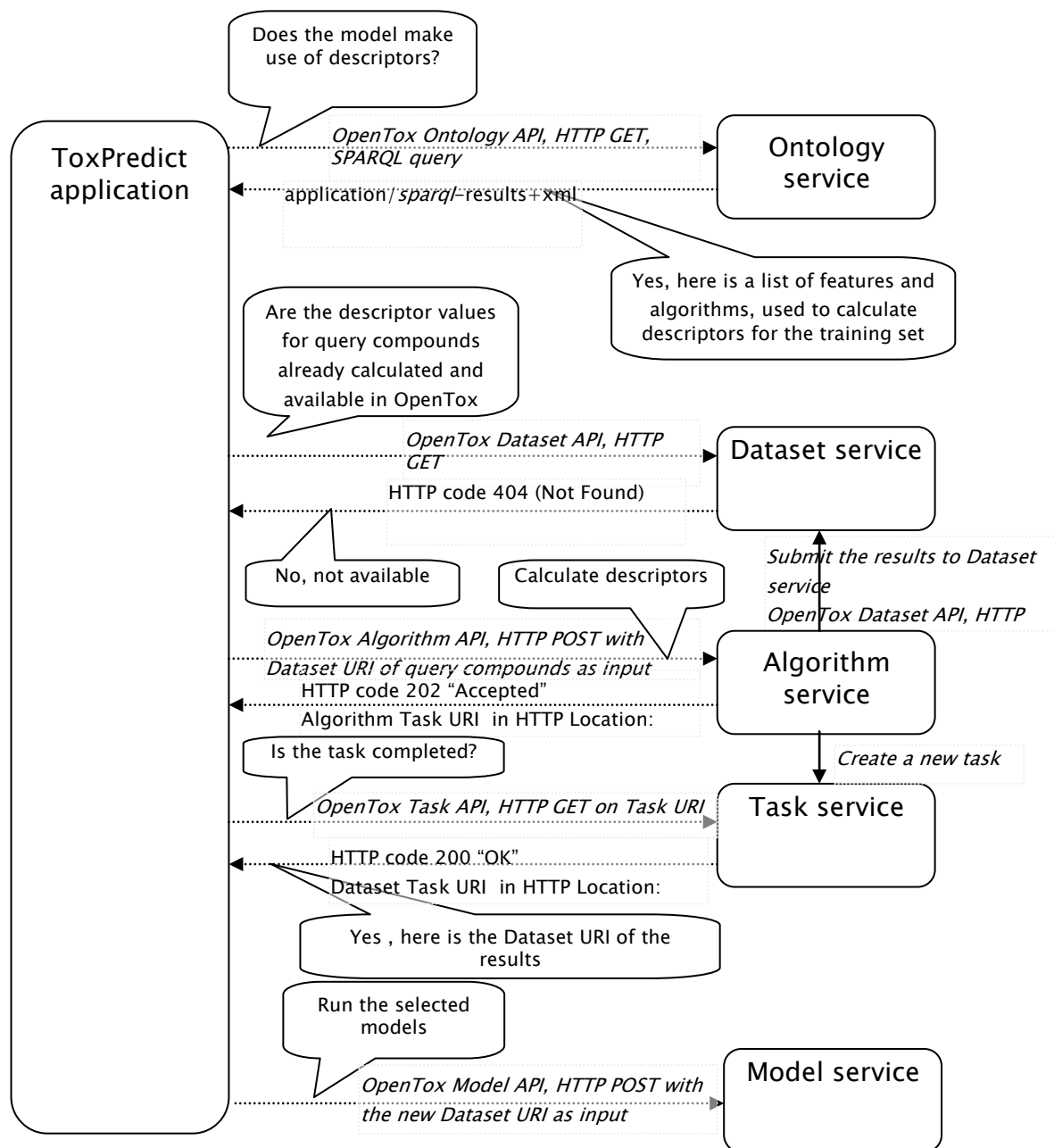
Step 3 – Behind the scenes – Previously, Algorithm, Model and Feature services had registered a list of algorithms, models and features into the Ontology service, by POSTing the URIs of these objects.



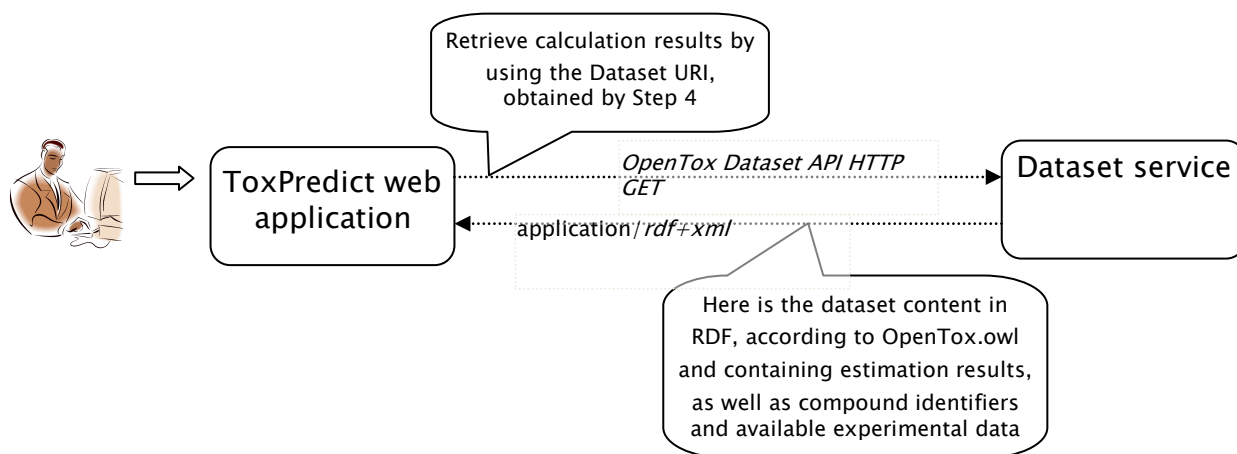
Step 4 – Model estimation



Step 4 – Behind the scenes



Step 5 – Display the results



4.2 ToxCreate Use Case

The **ToxCreate** use case, in contrast to **ToxPredict**, is aimed at researchers working in the life sciences and toxicology, QSAR experts, people interested in machine learning/statistics, pharmaceutical industry R&D and other related fields. It allows the creation of a number of models using one or more algorithms. Therefore it is not as easy to use as the **ToxPredict** application, as not only the algorithm has to be selected, but also the right parameter setting needs to be explored; these parameters are algorithm-dependent. For this decision-making, the expert has to have sound knowledge of the algorithm they are using.

The following sequence of screenshots, show a sample session of the **ToxCreate** use case.

1. Upload Dataset

ToxCreate

Create
Inspect
Predict
About

This service is for testing purposes only - once a week all models will be deleted. Please send bug reports and feature requests to our [issue tracker](#).

This service creates [lazar](#) classification models (more model building algorithms will follow) from your uploaded datasets. Here are [instructions](#) , for creating training datasets in Excel.

1. Enter a name for your endpoint:

2. Upload training data in [CSV](#) format:

[Cancel](#)

© [in silico toxicology](#) 2009-2010, powered by [OpenTox](#)

The first step of the ToxCreate workflow enables the user to specify a model training dataset in CSV format (this will be extended to other input means), consisting of chemical structures (SMILES) with binary class labels (e.g. active/inactive). The file is uploaded to the server and labelled with a user-defined name.

In contrast to ToxPredict, we here enable the user to specify his/her own training data/endpoint. This is done in batch mode, i.e. without interactive screens to select chemicals based on different criteria, which is convenient for expert users.

By hitting “Create model”, a QSAR model is derived. The current prototype demonstrates Lazar models only. No model parameters can be set at this time, but future versions will enable arbitrary OpenTox API-compliant models.

2. Create and Display Model

ToxCreate

Create
Inspect
Predict
About

This service is for testing purposes only - once a week all models will be deleted. Please send bug reports and feature requests to our [issue tracker](#).

Model creation started. Please be patient - model building may take up to several hours depending on the number and size of the input molecules.

Get an overview about ToxCreate models. This page is refreshed every 15 seconds to update the model status.

Hamster Carcinogenicity

Status:	completed (delete)
Started:	2010-02-26T13:07:57+01:00
Details:	85 compounds
URI:	http://webservices.in-silico.ch/test/model/49 (model representation in OWL-DL - experts only)
Validation:	to be added

This next screen in ToxCreate displays information about the model learned from the data submitted in the previous step. It features status information, date and number of compounds present in the dataset. A link leads to the complete specification of the model in OWL-DL. In the near future, it will be possible to validate the model by means of e.g. Cross-validation and select the most appropriate models for further evaluation.

At this point, the model is permanently stored on the server and can be used for predictions at any time in the future.

3. Select and Use Model(s) for Prediction

ToxCreate

Create
Inspect
Predict
About

This service is for testing purposes only - once a week all models will be deleted. Please send bug reports and feature requests to our [issue tracker](#).

Use this service to obtain predictions from OpenTox models.

Enter a compound identifier
Name, InChI, Smiles, CAS, ...

Choose one or more prediction models

Hamster Carcinogenicity	<input checked="" type="checkbox"/>
Hamster Carcinogenicity	<input type="checkbox"/>
mfg test	<input type="checkbox"/>
tcp1 bh1	<input type="checkbox"/>
MA-Sens	<input type="checkbox"/>
Andi	<input type="checkbox"/>
epafhm-120	<input type="checkbox"/>
test-vj-01	<input type="checkbox"/>
test bh 3	<input type="checkbox"/>
EPA Fathead Minnow Acute Toxicity	<input checked="" type="checkbox"/>
test bh	<input type="checkbox"/>

[Cancel](#)

© [in silico toxicology](#) 2009-2010, powered by [OpenTox](#)

In this step, a chemical (specified via SMILES code) can be entered in order to predict its chemical behaviour by arbitrary models existent on the server (note that in this way, in the future, arbitrary combinations of model algorithms and datasets/endpoints will be available to test the structure).

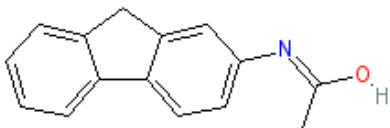
4. Display Prediction Results

ToxCreate

Create
Inspect
Predict
About

This service is for testing purposes only - once a week all models will be deleted. Please send bug reports and feature requests to our [issue tracker](#).

[New prediction](#)

C12C3=C(C=CC=C3)CC1=CC(=CC=2)NC(C)=O


**EPA Fathead Minnow
Acute Toxicity:**
active
(confidence: 0.150)

**Hamster
Carcinogenicity:**
active
(training data)

© [in silico toxicology](#) 2009-2010, powered by [OpenTox](#)

Step 4 displays the predictions made by the selected models from the previous step along with an image of the predicted structure. Based on the selections made in the previous screen, the expert user may predict the same structure by a variety of algorithms for the same dataset/endpoint and compare the predictions.

Conclusions

Together with model validation available from step 2, users will be able to select appropriate models with adjusted parameters beforehand. By predicting a variety of related endpoints, instead of just one, combined with arbitrary models at the same time, ToxCreate enables free modelling exploration along different dimensions.

4.3 Validation Use Case: Building and Validating a Model

Another important prototyped use case, besides the end user oriented applications described above (see section 4.1 and 4.2), is a training test set validation. This task can be executed using the validation web service prototype¹³ (developed at Albert Ludwig University (ALU-FR)) along with additional partner web services for algorithms, e.g., the Lazar and Fminer algorithms¹⁴ (provided by In Silicio Toxicology (IST)). These applications outline a successful implementation of the OpenTox API, and show interoperability of various web services, located at different locations. There is no graphical user interface provided yet, as this resource is to be executed by directly using REST operations. In the example described below, the command line tool curl¹⁵ is used. In the near future, validation routines like this will be fully integrated into the ToxCreate application.

Two validation examples follow in sections 4.3.1 and **Error! Reference source not found.** below. The first section outlines the validation workflow when validating the Lazar classification algorithm¹⁶ (provided by In Silicio Toxicology (IST)). The second section shows the evaluations of different regression models (provided by TUM) applied to a state-of-the-art QSAR dataset (provided by IDEA).

¹³ see OpenTox Deliverable 5.1. for more details

¹⁴ github.com/helma/opentox-algorithm

¹⁵ curl.haxx.se

¹⁷ Actually, a task object is returned first, while the process is running in the background. This is done for all time-consuming processes. We omit it in the description for simplicity.

4.3.1 Validating the Lazar algorithm

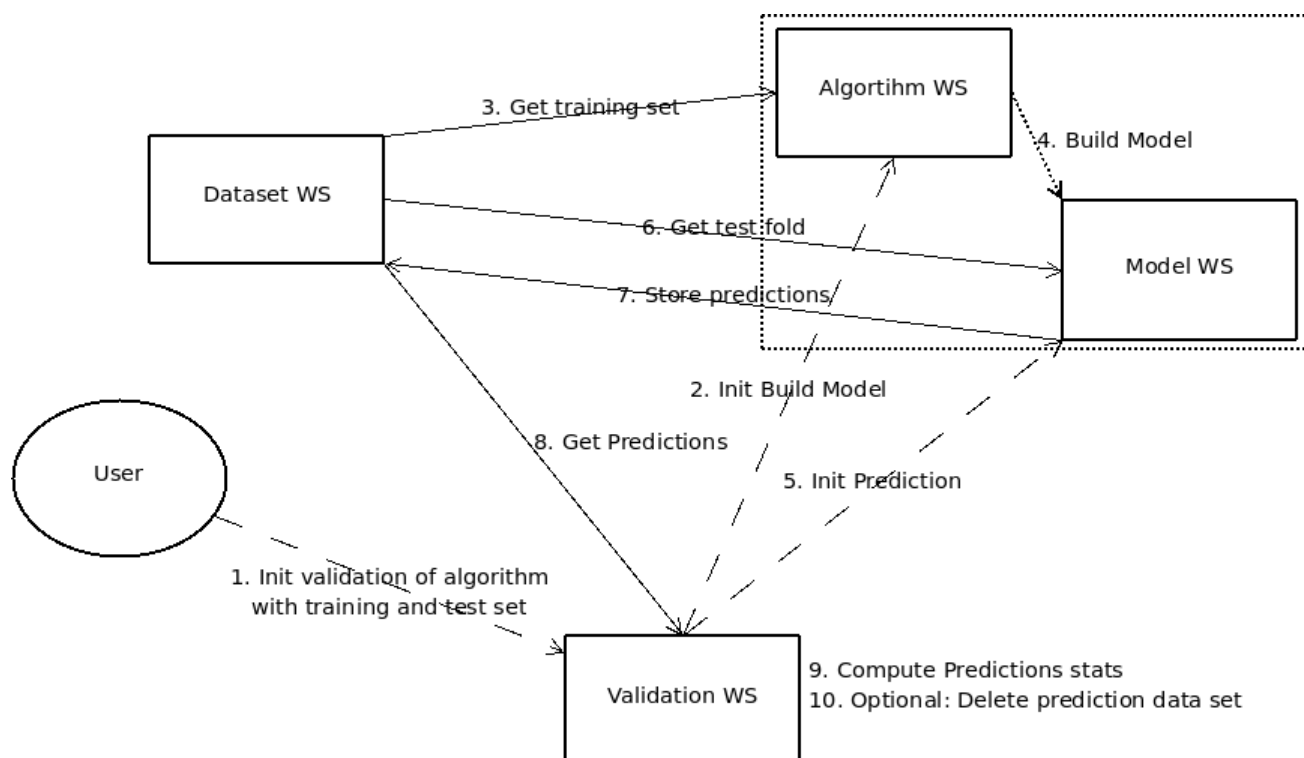


Figure 4.1: Workflow diagram illustrating the training test set validation of a prediction algorithm

The goal of this use case is to evaluate a prediction algorithm: the algorithm trains a model on a training dataset, and then predicts the compounds of a test dataset towards a certain endpoint. The validation result reflects how well the model performed. The workflow for the training test set validation is illustrated in Figure 4.1. Web services are displayed as rectangles; the three key POST REST operations are symbolized as dashed lines, while solid lines visualize data flow operations. The use case can be divided into 10 steps:

- Step 1: The user invokes the validation with a POST REST operation towards the validation web service, with parameters ‘algorithm_uri’, ‘training_dataset_uri’, ‘test_dataset_uri’, ‘prediction_feature’, and ‘algorithm_params’:

```
curl -X POST -d algorithm_uri=http://webservices.in-silico.ch/test/algorithm/lazar -d
training_dataset_uri=http://opentox.informatik.uni-freiburg.de/dataset/2 -d
test_dataset_uri=http://opentox.informatik.uni-freiburg.de/dataset/3 -d
prediction_feature=http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcome CPDBAS
Hamster -d algorithm_params="feature_generation_uri=http://webservices.in-silico.ch/test/algorithm/fminer http://opentox.informatik.uni-freiburg.de/validation
```

This is the only necessary REST call performed by the user. The subsequent steps are processed internally.

- Step 2: The validation web service starts the model building process, by automatically performing a POST REST operation addressed to the algorithm web service:

```
curl -X POST -d dataset_uri=http://opentox.informatik.uni-freiburg.de/dataset/2 -d
prediction_feature=http://www.epa.gov/NCCT/dsstox/CentralFieldDef.html#ActivityOutcome CPDBAS
Hamster -d feature_generation_uri=http://webservices.in-silico.ch/test/algorithm/fminer
http://webservices.in-silico.ch/test/algorithm/lazar
```

The algorithm web service fetches the training data and builds a model (Step 3 and 4). This use case includes a feature generation process that is omitted in the diagram: The Fminer service is used to mine structural features that occur in the compounds of the training dataset. The features are stored as a new dataset in the dataset web service, which is required to build the Lazar model. Finally, the algorithm web service returns the model URI as result¹⁷.

- Step 5: The validation web service uses the returned model URI to predict the compounds in the validation test set:

```
curl -X POST -d dataset_uri=http://opentox.informatik.uni-freiburg.de/dataset/3
http://webservices.in-silico.ch/test/model/<id>
```

The model web service therefore fetches the test compounds and makes predictions. The predictions are stored in a new dataset (Steps 6 and 7)

Steps 8 – 10: The validation web service retrieves the predictions from the dataset web service, and computes the validation statistics. In order to save hard-disk space, the prediction dataset can be deleted afterwards. This is an optional setting which is not included yet. Finally, the validation object is available to the user.

4.3.2 Validate regression models

The Fathead Minnow Acute Toxicity¹⁸ dataset is a well known dataset in the QSAR community, generated by the U.S. Environmental Protection Agency¹⁹. Russom et al²⁰ developed an expert system for this empirically-derived dataset to predict modes of actions from chemical structures.

We applied and evaluated different regression models, to see how well they could predict the LC₅₀ values that were experimentally determined for the chemical compounds in the

¹⁷ Actually, a task object is returned first, while the process is running in the background. This is done for all time-consuming processes. We omit it in the description for simplicity.

¹⁸ http://www.epa.gov/ncct/dsstox/sdf_epafhm.html

¹⁹ <http://www.epa.gov>

²⁰ http://www.epa.gov/nhrslup1/comptox/dsstox/Citations/ETC-Russom_1997_v16p948.pdf

dataset. Therefore, we computed 50 numerical descriptors using the computational chemistry library JOELIB²¹. The dataset has been randomly split into a training dataset²² including 389 structures, and a test dataset²³ with 193 structures. The datasets can be accessed at the ABMIT REST²⁴ web service, provided by partner IDEA. Partner TUM implemented web interfaces to various prediction algorithms²⁵ from the machine learning tool WEKA²⁶ from which we used a Nearest Neighbour algorithm for regression²⁷, M5P regression trees²⁸ and a Gaussian Processes learning algorithm²⁹.

As in the above section we performed a training test split validation, to first build a model on the training dataset, and then predict the LC₅₀ values of compounds in the training dataset:

```
curl -X POST -d algorithm_uri="http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/<regression_alogrithm>" \
-d training_dataset_uri="http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/639" \
-d test_dataset_uri="http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/640" \
-d prediction_feature="http://ambit.uni-plovdiv.bg:8080/ambit2/feature/264185" \
http://opentox.informatik.uni-freiburg.de/validation
```

The validation results of these operations show performance measures such as root mean square error and r^2 , providing indicators on the quality of the used models.

4.4 Testing Results

The testing of the web services and use cases is done on several different levels. The first level is the stress testing of the web services to identify potential performance decreases, the second level is manual API compliance testing of the web services, and finally the third and last level is the manual internal and external testing of the two application use cases.

²¹ <http://www.ra.cs.uni-tuebingen.de/software/joelib/introduction.html>

²² <http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/639>

²³ <http://ambit.uni-plovdiv.bg:8080/ambit2/dataset/640>

²⁴ <http://ambit.uni-plovdiv.bg:8080/ambit2/>

²⁵ <http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm>

²⁶ www.cs.waikato.ac.nz/ml/weka/

²⁷ <http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/kNNregression>

²⁸ <http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/M5P>

²⁹ <http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP>

4.4.1 Server Testing

For testing of the availability of the web services, the program SmokePing³⁰ has been set up. SmokePing allows keeping track of latency problems for individual web services and also allows email alerts to be sent in case of sudden server breakdowns. Figure 4.1 shows the graphical output for the IDEA web services.

Using this availability testing, we have already identified some performance problems with the web services at ALU-FR and IST as well with SL's server network connectivity. The performance problems for ALU-FR and IST are currently being resolved. To enable SL a proper web service, hosting provided by ALU-FR for SL services will be addressed within the next development iteration.

SL's server network connectivity was suffering from high latency and substantial packet loss (up to 60%), the network connectivity has to be either radically improved by some means or (if this proves to be difficult or even impossible), then any software, developed by SL, should be deployed in a more network-friendly location, e.g. somewhere in Europe. In December 2009, ALU-FR has given SL access to a dedicated server in Germany.

³⁰ oss.oetiker.ch/smokeping/index.en.html

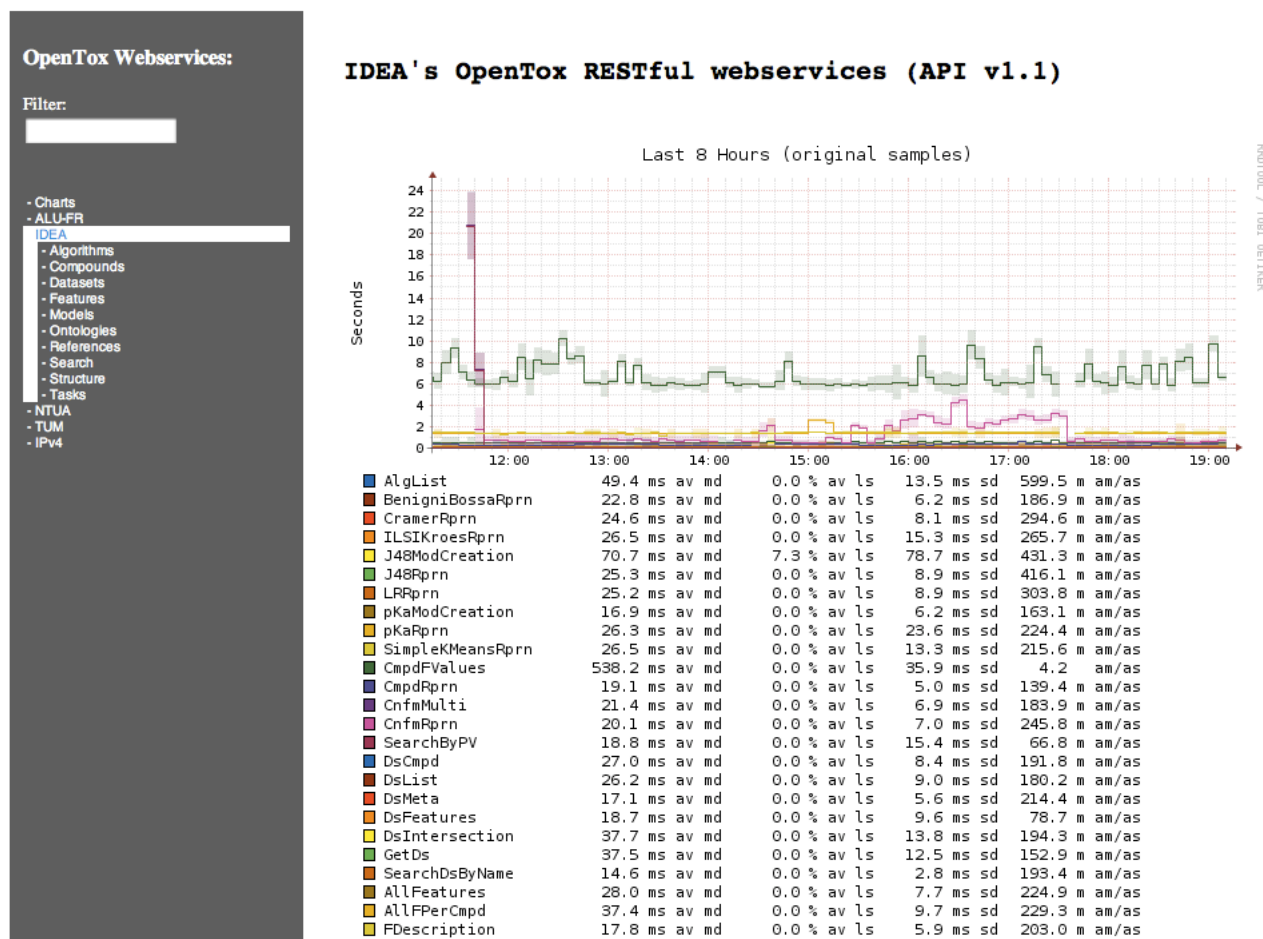


Figure 4.1 An example output of SomePing for IDEAs webservices³¹

4.4.2 API testing

In December 2009, we performed a web service API test. The test showed a general lack of API compliance and interoperability. However, having performed this test, we could prioritize on web services development and have already addressed several of the problems identified.

In the following list, we present the major findings of the test in December and the counter measures taken or state what the current status within the OpenTox Framework is.

³¹ ambit.uni-plovdiv.bg/cgi-bin/smokeping.cgi?target=IDEA

- A number of issues of various severity in practically all services and implementations, which have been identified were reported to the respective developers and partially or fully resolved;
- Interoperability between different implementations had been successfully demonstrated in one case only in December 2009. Now, in March 2010, more services are able to interoperate, such as IDEA, TUM, and NTUA as well as between ALU-FR and IST. Interoperability between the validation web services (ALU-FR) and the IDEA, TUM, and NTUA services has also been established in a couple of cases;
- At the time, the model building procedures seemed to be quite fragile, subject to frequent failures, lack of interoperability, and/or were not fully API 1.1-compliant. These issues have been addressed and have been partially resolved;
- IDEA's implementation covers the largest API subset so far and exhibits reasonable interoperability between Compound, Feature, Dataset, Algorithm, Model and Task services in the framework of a single implementation;
- The implementation of the Validation service is ongoing at ALU-FR and more or less in sync with other developments;
- IBMC's OpenTox service implementation was left out of the SmokePing measurements, because it was not API 1.1 compliant. This issue has not yet been resolved.

To summarize, we are continuously and rigorously testing the interoperability between the individual OpenTox web services as well as API compliance.

4.4.3 Use case testing

Our prototype use cases, ToxModel and ToxPredict, were tested and evaluated internally as well as externally by a third party. The internal testing was performed twice, as the testing started at a very early stage of the two use cases being somewhat operational. The second internal test was then held one week later, where already some of the issues were resolved (see below). The external testing was performed on the 23rd of February 2010. To document the results of all tests, we designed a *Use Case Beta Testing Report Template* (see 6. Appendix). Using this template, testers can fill in their experience with the prototype with regards to functionality, usability, and design and allow furthermore for feedback with regards to possible improvements and bug reporting.

ToxPredict

The internal tester had several issues with the prototype, which we present here as well as measures to address the issues raised:

- *Structures not found and no proper message displayed.*
 - This is currently being fixed and will be available first week of March 2010
- *File upload was missing during testing session.*
 - This is available since Feb 26 2010
- *No explanation of reliability of estimates in final report.*
 - This is hard to provide, without integration with the validation service, applicability domain and mechanistic interpretation by human expert. These steps are planned as next developments.
- *Initial version, tested internally was providing endpoint and model selection via hierarchy of endpoints.*
 - This seems to be inconvenient for the user and was replaced with a flat list of models and related endpoints.
- *Result reports are provided via different formats (SDF, PDF, CSV) and accessible via small icons.*
 - This seems to be not always intuitive and will be redesigned.
- The formatting and content of the result reports need to be synchronized with REACH-related report formats such as QMRF and Chemical Safety Assessment.

Within the external testing the following issues were raised (for clarity we include the actual question from the questionnaire):

- *Input chemical structure as compound name: "What is the naming convention used?"*
 - This is free text search, based on multiple synonyms from multiple sources, available in the database. We are currently working on integration of further details and guidance in the ToxPredict GUI.
- *Input chemical structure as CAS number: "No issues experienced but would be helpful to know the scope of the inventory underlying the CAS database and the extent to which it has been curated"*

ECHA pre-registration list; curation explained in OpenTox D3.3 report. We will be subsequently introducing this information into the ToxPredict GUI.
- *Input chemical structure as arbitrary string: "Not sure what this means?"*
 - Free text search (guidance is being added).
- *Verify selected structure(s): "But what purpose does this serve? If there were some editing functionality to modify a structure rather than clicking back and having to start over then that would be useful"*

- This is due to yet unimplemented functionality, and will allow to select/deselect structures, as well as launching structure diagram editor to modify structures, if appropriate.
- *Select a relevant model from a list of available models: “No supporting information of the models. The excel, PDF icons are confusing – could they be omitted until you reach the display results section?”*
 - Excel and PDF icons provide exactly supporting information – e.g. training dataset. Including more information about models is under development – this already exists in the corresponding web services, but is not yet exposed via the ToxPredict front end. Redesign of the page will be considered as well.
- *Apply model(s) (make a prediction(s)): “Can’t go back and re-select some more models, you have to start the whole process again.”*
 - Already fixed in current version.
- *Apply model(s) (make a prediction(s)): “What is the OpenTox model – there is no endpoint associated with it?”*
 - During the testing session, it was a test model, generated by OpenTox TUM services, which has not been yet assigned an entry from the endpoint ontology and was serving only as a demo for integration between services, developed by different partners and running on remote locations. We will ensure that sufficient details are available in future for every model presented in the ToxPredict GUI.
- *Comment: “Software was easy to apply to my specific situation, but at present does not offer any additional benefits over using the software as standalone”*
 - The demo software includes mostly ToxTree models, with additional ones as *pKa* estimation and the demo regression model (TUM) for aquatic toxicity, thus the perception is not much different than ToxTree, for the time being. However, the software technology used differs drastically and allows inclusion of models, which are available from several partners, on remote locations and by different software technologies. This was not sufficiently demonstrated and not evident for the tester. We will need to design a demo case shortly to demonstrate the availability and advantages of integration of different models.

Overall, we also found that the user would like to have a progress bar during the time of computation, including the possibility to stop a process. We are currently considering an API extension to allow such functionality. Furthermore, the reporting format and content display need improvement.

ToxCreate

This use case has also been tested twice internally and once externally. At the time of testing, the ToxCreate prototype was still in an early stage and the external tester found that it was hard to judge this workflow, as at the time only very limited functionality was available. However, the general impression was “General idea is good but implementation and functionality are too embryonic. Expectations are not met”.

As with the internal tester, some issues were raised and are being addressed:

- *Train a model: “I made a copy of the dataset with some deliberately corrupted fields, and ran through the same sequence, but the results were the same as before with no mention of my deliberate data errors”*
 - This has been addressed and resolved.
- *Train a model: “I uploaded the Hamster test file CSV file with a few corrupted data values (a few numbers changed to letters). On clicking “Create Model”, it simply returned to same page saying upload CSV file (ideally, it should provide a more explicit error message, or continue and ignore the erroneous values).*
 - We are addressing the problem with missing error messages and warnings as well as improving the online documentation.
- *Train a model: “I made a copy of the dataset having deleted all except one single calibration compound, and ran through the same sequence, but the results were the same as before with no mention of any errors.”*
 - This has been addressed. Now a warning appears if only a few training compounds are supplied.
- *Train Model: “I clicked “Create Model”, the page immediately changed to the “Predict” page, but there was no indication if anything had happened or if it was still calculating (e.g. no spinning cursor).”*
 - This has been addressed and a message is prompted that the process has been started. This page is then refreshed every 15 seconds until the completed message appears.
- *Make predictions: “When I entered a smiles string into the “Enter a compound identifier” box and clicked “Predict”, a picture of the correct structure appeared with a similar error message as above <<test6: not available>>.”*
 - This has been resolved. Currently the user has to select an available model, enter a SMILES string and is able to predict the activity for the entered compound using the just created model.

- *Lack of SDF file support*
 - We will resolve this at a later stage of the development cycle, using a Ruby – OpenBabel SDF converter. However, currently the Ruby bindings for OpenBabel are not fully operational.

Testing Summary

At the time of the first internal and external tests, functionality was still very limited and documentation still missing. However, with these use cases, some interoperability has been achieved and is ongoing work. We are planning to have additional alpha and beta testing employing a number of third parties during Spring 2010.

5. Discussion

5.1 Further Working Directions

In future OpenTox developments we will extend the OpenTox Framework substantially by:

- Achieving improved and complete interoperability between all OpenTox web services and specifically within and between the ToxPredict and ToxCreate applications;
- Extension of the Framework to incorporate models and data related to data mining and *in vitro* assays probing mechanistic pathways;
- Extension of the Framework to incorporating biological models and data related to probing kinetics and exposure prediction;
- Provision of additional graphical user interfaces and applications, by incorporating the results of continuous internal and external testing procedures;
- Extending the employed ontologies to align with current standards such as the Blue Obelisk and OBO Foundry ontologies, allowing the full description of predictive toxicology algorithms, including references, parameters and default values;
- Addition of QMRF reporting facilities, by producing pre-filled skeleton QMRF reports, which are then editable within the QMRF Editor³²;

³² <http://ambit.sourceforge.net/qmrf/jws/qmrfeditor.jnlp>

- Integration of authentication and authorization into the Framework, allowing for confidential data to be integrated as well as allowing restricted access to certain datasets;
- Organisation of workshops, seminars and tutorials to communicate the OpenTox Framework to other interested parties, and offering algorithm developers as well as toxicological risk assessors the possibility to participate in the community approach. (For example, a one day workshop “Development of Predictive Toxicology Applications” will be held alongside the EuroQSAR 2010 conference in Rhodes in September 2010:
www.opentox.org/data/blogentries/public/opentoxworkshoprhodes2010)

5.2 Conclusions

The OpenTox Framework offers a standardized interface to state-of-the art predictive toxicology algorithms, models, datasets, validation and reporting facilities on the basis of RESTful web services³³ and guided by the OECD Principles, REACH legislation and user requirements.

The Framework supports rapid application development and extensibility by using well-defined ontologies, allowing simplified communication between individual components. Two user-centered prototype applications, ToxCreate and ToxPredict, show the potential impact of the framework regarding high-quality and consistent structure-activity relationship modeling of REACH relevant endpoints. The applications have been made available publicly on the Web (www.opentox.org/toxicity-prediction) providing immediate access to the applications as they have been developed.

ToxPredict satisfies a common and important situation for a user wishing to evaluate the toxicity of a chemical structure. The user does not have to cope with many current challenges such as the difficulty of finding or using existing data or the complications of creating and using complicated computer models. Because of the extensible nature of the standardised design of the OpenTox Framework, many new datasets and models from other researchers may be easily incorporated in the future, both strengthening the value offered to the user and ensuring that research results are not left languishing unused in some isolated resource not accessible to the user. The approach offers the potential to be extended to the complete and easy-to-use generation of reporting information on all

³³ Fielding, R.T., *Architectural Styles and the Design of Network-based Software Architectures*, Ph.D. dissertation, in University of California, Irvine. 2000

REACH-relevant endpoints based on existing available scientific research results, and indications when additional experimental work is required, thus satisfying currently unmet industry and regulatory needs.

ToxCreate provides a resource to modellers to build soundly-based predictive toxicology models, basely solely on a user-provided input toxicology dataset that can be uploaded through a web browser. The models can be built and validated in an automated and scientifically sound manner, so as to ensure that the predictive capabilities and limitations of the models can be examined and understood clearly. Models can subsequently be easily made available to other researchers and combined seamlessly into other applications through the OpenTox Framework.

6. Appendix

Example Beta Testing Report Template

1 General Instructions

Please complete the ToxPredict Beta Test Tasks described below. To run the ToxPredict software you would need a web browser (a recent version of Firefox or Internet Explorer) and a network connection to Internet. Please answer the questions on the attached form, either by hard copy, or by editing an electronic copy of this document. Please return your feedback to Vedrin Jeliaskov vedrin.jeliaskov@gmail.com. With your permission, we may contact you occasionally during the course of the beta testing to solicit interim feedback. You might also want to register at the OpenTox site³⁴ and provide further feedback through the test case development issue tracker³⁵.

The ToxPredict software implements a prototype use case of the OpenTox framework, which enables end users to run existing endpoint-specific models on a given compound (or dataset) and get model predictions. The main steps of the workflow are as follows:

1. Select input compound (enter chemical name, registry identifier (e.g. CAS, EINECS), SMILES, InChI, arbitrary keyword, SMARTS or draw molecule in molecular editor);
2. Select specific endpoint (e.g. Human Health Effects / Carcinogenicity);
3. Select one or more models, available for this particular endpoint (e.g. ToxTree: Benigni/Bossa rules for carcinogenicity and mutagenicity);
4. Apply selected model(s);
5. View and/or retrieve the resulting report, available in various formats, e.g. HTML, SDF, CML, SMI, PDF, XLS, ARFF or RDF.

2 Beta Testing Objectives

The main objectives of this beta testing exercise are:

- To evaluate ToxPredict' technical capabilities and scientific value;
- To evaluate ToxPredict' ease of use and interactivity;
- To evaluate the end user documentation;
- To identify errors/bugs;
- To compile and prioritise a wish list of missing features, to be implemented in subsequent versions of the OpenTox framework.

3 Beta Testing Tasks

1. Complete **Part-A: Identification** (provide your name and contact details, web browser type/version and time period when the testing has been performed).

³⁴ www.opentox.org/join_form

³⁵ www.opentox.org/dev/testing/testcasedevelopment/testcasedevelopmentissuetracker

2. Open the following URL in your web browser <http://93.123.36.100:8180/ToxPredict>
3. Proceed with functional evaluation of ToxPredict by following as many variants of the provided workflow as possible. These activities aim to evaluate the software's basic ability to generate the expected results, in the way you need them. Report your findings in **Part-B: Functional Evaluation**.
4. Complete **Part-C: Overall Comments and Usability Evaluation**. This section asks you to rate various aspects of the software using a 5-point scale.
5. List any bugs or problems in **Part-D: Specific Bugs and Problems Noted** as you proceed.
6. Please answer any other relevant questions listed in **Part-E: Other Generic Topics**.

4 Known ToxPredict Problems

1. Bugs/usability problems:
 - a. Workflow navigation doesn't work always as expected and is subject to improvement;
 - b. The overall GUI design is subject to improvement.
2. Missing features:
 - a. The integrated online help doesn't provide sufficiently detailed guidance;
 - b. Support for batch processing of datasets is under development;
 - c. Support for file upload is under development;
 - d. Support for molecular structure drawing is under development;
 - e. Support for SMARTS searching is under development;
 - f. Integrated descriptor calculation is under development;
 - g. Model integration is under development (only ToxTree and pKa models are fully supported at the time of this writing);
 - h. Models are available only for a subset of endpoints.

5 Part-A: Identification

Your Name	
Your Organisation	
Your Phone number	
Your E-mail address	
Used web browser (type/version)	
Time period when the testing has been performed	

6 Part-B: Functional Evaluation

Test Case ID	Function	Tested? (yes/no)	Comments, Ideas and Issues
ToxPredict -01	Input chemical structure as SMILES		
ToxPredict -02	Input chemical structure as MOL		
ToxPredict -03	Input chemical structure as SDF		
ToxPredict -04	Input chemical structure as InChI		
ToxPredict -05	Input chemical structure as compound name		
ToxPredict -06	Input chemical structure as CAS number		
ToxPredict -07	Input chemical structure as EINECS number		
ToxPredict -08	Input chemical structure as SMARTS		
ToxPredict -09	Input chemical structure as arbitrary string		
ToxPredict -10	Input chemical structure through the integrated molecular structure editor		
ToxPredict -11	Select an endpoint from a list of available endpoints		
ToxPredict -12	Select a relevant model from a list of available models for a given endpoint		
ToxPredict -13	Apply model(s) (make a prediction(s))		
ToxPredict -14	Follow the progress of a prediction task		
ToxPredict -15	View predictions and experimental data (HTML format)		
ToxPredict -16	Retrieve resulting report in SDF format		
ToxPredict -17	Retrieve resulting report in CML format		
ToxPredict -18	Retrieve resulting report in SMI format		
ToxPredict -19	Retrieve resulting report in PDF format		
ToxPredict -20	Retrieve resulting report in CSV format		

Test Case ID	Function	Tested? (yes/no)	Comments, Ideas and Issues
ToxPredict -21	Retrieve resulting report in ARFF format		
ToxPredict -22	Retrieve resulting report in RDF format		

7 Part-C: Overall Comments and Usability Evaluation

Usability Question	Rating Scale	Specific Comments on Rating
	1 – Strongly Disagree 2 – Somewhat Disagree 3 – Neither Agree, Nor Disagree (No Opinion) 4 – Somewhat Agree 5 – Strongly Agree	
Overall		
This software is useful to me now, or it will be in the near future		
System output and visualization are useful and meet my needs		
Software has the capabilities I need (note any exceptions)		
General impression is good (why?)		
Software was easy to apply to my specific situation		
Data entry effort is manageable		
Technical Content		
Appropriate technical and scientific basis is used		
Uses proper terminology		
Performs calculations correctly		
Toolbars, menus, commands and options are appropriate		
Labels and terms are accurate and easy to understand (if not, what would you prefer?)		
Data formats are useful (if not, what would you prefer?)		
I entered my own data and received the expected results		
Boundary values (largest and smallest chemical samples) were handled correctly		

Usability Question	Rating Scale	Specific Comments on Rating
	1 – Strongly Disagree 2 – Somewhat Disagree 3 – Neither Agree, Nor Disagree (No Opinion) 4 – Somewhat Agree 5 – Strongly Agree	
Software Operation		
Trouble-free operation		
Easy to navigate within the software		
Consistent and logical flow in using the software		
Easy to find what you are looking for		
Software works as expected (uses standard user interface features)		
Software works well within its family of software applications (if known)		
Files import and export to other needed applications		
Prints properly to a printer		
Documentation		
Clearly describes software purpose		
Organization is clear and logical		
Examples show how to use the main features (please list any features needing more explanation or examples)		
Tables, graphs & figures provide sufficient guidance through major software options		
Do error messages clearly direct the user to a solution?		
On-line help: was it easy to find what you wanted?		
Included necessary technical support information		
Appearance		
Colours, symbols, and graphics are legible and pleasing		
Looks professional		
Correct spelling & grammar		
Application windows have consistent look and feel		

8 Part-D: Specific Bugs and Problems Noted

[illegible]

9 Part-E: Other Generic Topics

Please comment on the following (if relevant):

- scientific value of algorithms included
- speed of user interface interactivity and of calculations
- order of screens and steps, and number of steps to complete an action
- compatibility of the software with existing workflows
- organization of menu items
- quality of written explanations
- terms or abbreviations used
- annoying or frustrating experiences