Deliverable D2.3

Prototype Demonstration Server

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	Prototype Demonstration Server	
Document ID:	OpenTox Deliverable Report 2.3	
Date:	Sept 30, 2010	
Status:	Final Version	
Organisation:	Technische Universität München (TUM)	
Contributors	Stefan Kramer (SK)	TUM
	Tobias Girschick (TG)	TUM
	Fabian Buchwald (FB)	TUM
	Vedrin Jeliazkov (VJ)	IDEA
	Martin Gütlein (MG)	ALU
	Jörg Wicker (JW)	TUM
	Nina Jeliazkova (NJ)	IDEA
	Barry Hardy (DC)	DC

Distribution:	Public

Purpose of Document:	To disseminate results on the initial OpenTox Prototype

Document History:	1 – First version of report (SK, FB, TG) on Aug 16
	2 – Reviewed and edited by VJ on Aug 24, 2010
	3 – Chapter six first version by TG on Aug 27, 2010
	4 – Created 3.3 (ALU technical realization)
	5 – Added Taverna section; TG and JW; Aug 31, 2010
	6 – NJ added section 4.4; Aug 31, 2010
	7- BH (DC) reviewed and edited final version

Table of Contents

Acknowledgements

Research Funding

OpenTox - An Open Source Predictive Toxicology Framework, www.opentox.org, is funded under the EU Seventh Framework Program: HEALTH-2007-1.3-3 Promotion, development, validation, acceptance and implementation of QSARs (Quantitative Structure-Activity Relationships) for toxicology, Project Reference Number Health-F5-2008-200787 (2008-2011).

Project Partners

Douglas Connect (DC), In Silico Toxicology (IST), Ideaconsult (IDEA), Istituto Superiore di Sanita' (ISS), Technical University of Munich (TUM), Albert Ludwigs University Freiburg (ALU), National Technical University of Athens (NTUA), David Gallagher (DG), Institute of Biomedical Chemistry of the Russian Academy of Medical Sciences (IBMC), Seascape Learning (SL), Jawaharlal Nehru University (JNU), Fraunhofer Institute for Toxicology & Experimental Medicine (ITEM).

Advisory Board

European Centre for the Validation of Alternative Methods, European Joint Research Centre, U.S Environmental Protection Agency, U.S. Food & Drug Administration, Nestlé, Roche, AstraZeneca, Lhasa, Leadscope, University of North Carolina, Pharmatrope, Bioclipse, EC Environment Directorate General, Organisation for Economic Co-operation & Development, CADASTER, Bayer Healthcare.

Correspondence

Dr. Barry Hardy, OpenTox Project Coordinator and Director, Community of Practice & Research Activities, Douglas Connect, Baermeggenweg 14, 4314 Zeiningen, Switzerland

Email: barry.hardy -(at)- douglasconnect.com

Summary

OpenTox provides an interoperable, standards-based framework for the support of predictive toxicology data management, algorithms, modelling, validation and reporting. OpenTox provides end-user oriented tools to non-computational specialists, risk assessors, and toxicological experts in addition to Application Programming Interfaces (APIs) for developers of new applications.

OpenTox includes services for compounds, datasets, features, algorithms, models, ontologies, tasks, validation, and reporting which may be combined into multiple applications satisfying a variety of different user needs. OpenTox applications are based on a set of distributed, interoperable OpenTox API-compliant REST web services. The services are operated on a network of publicly accessible prototype demonstration servers that are distributed over multiple locations and organisations. Every provider responsible for service contributions has a server running that contributes to the distributed OpenTox system. In this report we take a closer look at three example server implementations to describe the whole distributed prototype system. The report also provides insight into the technical details, e.g., what programming languages and technologies have been used. It provides example usages of the distributed system, so that potential contributors to the system have a key overview guidance at their hands that will, in addition to further online information on the OpenTox web resources (www.opentox.org), enable easy access to the system itself and make contributions easier.

Two initial OpenTox applications were prototyped as an illustration of the potential impact of OpenTox for high-quality and consistent structure-activity relationship modelling of REACH-relevant endpoints: ToxPredict which predicts and reports on toxicities for endpoints for an input chemical structure, and ToxCreate which builds and validates a predictive toxicity model based on an input toxicology dataset. Because of the extensible nature of the standardised framework design, barriers of interoperability between applications and content are removed, as the user may combine data, models and validation from multiple sources in a dependable and time-effective way.

Taverna (www.taverna.org.uk) has been used to provide a user friendly workflow system to access and combine OpenTox web services. It provides a user interface which can be used to generate arbitrary workflows from combinations of single OpenTox web services. The web services are combined by importing single web services and connecting their inputs and outputs using a point-and-click user interface.

Continuing effort will be carried out by OpenTox developers to meet current academic and industry challenges regarding interoperability of software components and integration of data, algorithm and model services within the context of tested Use Cases. The experience we have gained during this work should help speed up the development process towards this direction.

1. Introduction

Basic OpenTox [HAR10] design principles are the interoperability, flexibility and extensibility of the produced software. To comply with those principles, we decided to use the REST-based [FIE00] web service technology for the OpenTox framework component implementation. The REST web service technology is a young, lightweight, emerging technology. It has been adopted by internet industry giants Google and Amazon who both provide REST Application Programming Interfaces (APIs) to access services. Furthermore, OpenTox partners decided to initially build a distributed system of web services to achieve maximum flexibility and extensibility. As a consequence, there is no such thing as one public prototype demonstration server for OpenTox, but the concept is a network of publicly accessible prototype demonstration servers that are distributed over the partner locations. Basically, every provider responsible for service contributions has a server running that contributes to the distributed OpenTox system. In this document we take a closer look at three (IDEA, ALU and TUM) example server implementations to describe the whole distributed prototype system. The three servers are sufficient to explain the conceptual functionality and interactions of the prototype system as they together

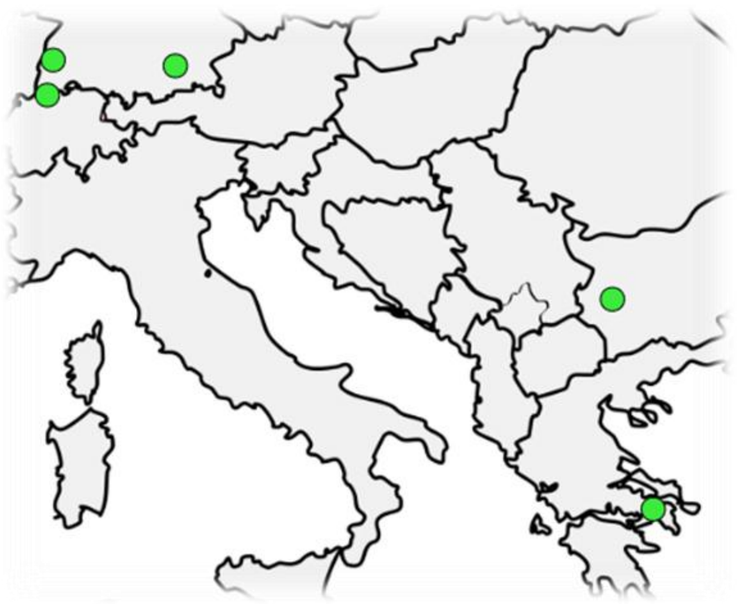


Figure 1 Map of distributed OpenTox servers

implement all components of the OpenTox API. Other up and running servers are hosted by NTUA, JNU and IST.

The document also provides insight into the technical details, e.g., what programming languages and technologies have been used. It provides example usages of the distributed system, so that potential contributors to the system have an overview guidance at their hands that will, in addition to further online information on the OpenTox web resources (www.opentox.org), enable easy access to the system itself and make contributions easier.

2. System overview

Prototype implementation of the OpenTox framework consists of different distributed components or servers that interact and work together in a REST-based web service environment. In this report focus on the three servers of IDEA, TUM and ALU that are sufficient to explain the conceptual functionality of the OpenTox framework prototype system; others of note include the servers of NTUA and IST. The three servers focused on include all components of the OpenTox API as well as implementations based on different technical concepts and programming languages. IDEA's server that is located at ambit.uni-plovdiv.bg:8080/ambit2 or apps.ideaconsult.net:8080/ambit2 is primarily a compound or dataset server that provides unified access to compounds, features including endpoints and complete datasets. Besides this, *in silico alert* methods based on Toxtree are provided. TUM's server (opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/) provides different algorithms, ranging from descriptor generation and descriptor selection to standard machine learning and data mining algorithms. Also provided are several methods that are implemented particularly for the needs of toxicology predictions, QSAR model learning or descriptor generation, e.g., FMiner [MAU10], FCDE [BUC10], Toxtree [PAT08] and lazar [HEL06]. Using IDEA's and TUM's server, the retrieval of chemical information, descriptor generation, descriptor selection, model learning and model application is possible. For validating a model and for the generation of a report of predictions, communications with ALU's server (opentox.informatik.uni-freiburg.de), which provides standard validation routines and reporting tools, is necessary. Other OpenTox servers provide additional data and methods, i.e. additional complexity in the form of QSAR or data mining algorithms: NTUA (opentox.ntua.gr), JNU/SL (opentox2.informatik.uni-freiburg.de:8080/MaxtoxTest) and IST (webservices.in-silico.ch).

3. Technical Realization

The selection of the REST-based distributed web service architecture enables the flexible integration of existing tools that are available in different programming languages as well as on different platforms. Consequently, the technical realization of a new web service is possible in the programming language of choice, so long as REST interfaces are available. Currently Java (java.sun.com) and Ruby (www.ruby-lang.org/en/) are the main languages used. We provide an overview here of three example servers.

3.1 Services at TUM

The TUM services are all implemented in the Java programming language and can be tested with a stand-alone Java server or in a Java servlet container. A java example for TUM's implementation of a regression algorithm is provided in Appendix A. The production environment is an Apache Tomcat servlet container. TUM developers also have used the resin web-container as an alternative to the Tomcat server. Basically every Java servlet container should be usable. Only some underlying non-Java software remains in native code, e.g. the Free Tree Miner software remains in C++ code that is called from within Java. This means that old code is reusable very quickly and must not be transferred or re-implemented. To avoid redundant and time-consuming calculations with frequency-based descriptor calculation algorithms, the TUM service checks, whether FTM or gSpan calculations have already been done before, by saving algorithm configurations into a database. This is either an SQLite database, if no PostgreSQL is installed on the server, or a PostgreSQL database. For providing the REST-based web service functionality the TUM service uses Java restlet classes (www.restlet.org). For the ontology-related parts of the code (RDF/OWL) the service uses the JENA java packages (jena.sourceforge.net).

3.2 Services at IDEA

IDEA's web services are also implemented in the Java programming language and can be run in any servlet container. Two independent instances have been deployed and tested in Apache Tomcat (tomcat.apache.org) version 6.0.29:

- Development instance (ambit.uni-plovdiv.bg:8080/ambit2/ ambit.uni-plovdiv.bg:8082/ontology/), running on Linux ambit 2.6.26-2-amd64 #1 SMP Sun Jun 20 20:16:30 UTC 2010 x86_64 GNU/Linux
- Production instance (apps.ideaconsult.net:8080/ambit2/ apps.ideaconsult.net:8080/ontology/), running on FreeBSD eos.ideaconsult.net 8.0-RELEASE-p3 FreeBSD 8.0-RELEASE-p3 #0: Wed Jun 2 23:56:19 EEST 2010 root@eos.ideaconsult.net:/usr/obj/usr/src/sys/EOS amd64
- MySQL (www.mysql.com) is used as a database backend. Continuous availability and performance monitoring of the production instance is performed with statistics available online at ambit.uni-plovdiv.bg/cgi-bin/smokeping.cgi?target=IDEA

3.3 Services at ALU

The services from ALU are implemented with the Ruby programming language, based on the web application framework Sinatra (www.sinatrarb.com). The current validation web service runs with an Apache HTTP web server (httpd.apache.org). Various other web servers such as nginx (nginx.org) and Thin (code.macournoyer.com/thin/) have been successfully tested during development.

ALU and IST develop a ruby library (github.com/mguetlein/opentox-ruby-api-wrapper) to simplify access to different OpenTox web resources via Ruby objects.

A MySQL database server is used to store data. Further software used by the validation reporting services is Gnuplot (www.gnuplot.info) and Saxon (wiki.docbook.org/topic/Saxon).

To guarantee an easy installation of all necessary components, ALU and IST offer a complete appliance that can be run in Oracle (formerly SUN) VirtualBox (wiki.github.com/helma/opentox-documentation/installation-of-opentox-webservices).

4. Service Examples

This section gives examples of the prototype system's functionality using cURL command line calls and code examples.

4.1 cURL as a Quick-check Tool

A lot of quick checks and examples in the OpenTox community use the open software tool cURL (curl.haxx.se). cURL is a command line tool for transferring data with URL syntax, supporting FTP, FTPS, HTTP, HTTPS and many other protocols as well as HTTP form based upload, proxies, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, kerberos...) and a lot of other useful tricks. All offered web services in the OpenTox framework can be addressed with cURL so examples in this document and in the API (opentox.org/dev/apis) are often given with cURL calls.

4.2 Dataset Querying and Merging

In this example we show how to query for a dataset and how to merge different sources of mutagenicity data into one dataset. The first dataset is a benchmark dataset for *in silico* prediction of Ames mutagenicity. Its URI is apps.ideaconsult.net:8080/ambit2/dataset/2344. Meta-information on the dataset can be retrieved via apps.ideaconsult.net:8080/ambit2/dataset/2344/metadata. The easiest way to access the addresses is to copy and paste the URLs to the address field of a web browser. If the activity column (apps.ideaconsult.net:8080/ambit2/feature/28958) is selected, we get information on the selected feature. Now the question we ask in this example is: Is there other mutagenicity data available that one can use?

This can either be done via the given link in the "Same as" column of the activity feature that initiates a search for data with the same EChA endpoint (in this case mutagenicity), or

directly via a URL that represents the search (mark that special characters in the search parameter *sameas* have to be replaced):
http://apps.ideaconsult.net:8080/ambit2/feature?sameas=http%3A%2F%2Fwww.opentox.org%2FechaEndpoints.owl%23Mutagenicity

Now merging the information of the different mutagenicity datasets is done by just plugging together the relevant feature URIs and the dataset URI in the following way:
http://apps.ideaconsult.net:8080/ambit2/dataset/2344?feature_uris[]=http://apps.ideaconsult.net:8080/ambit2/feature/28958&feature_uris[]=http://apps.ideaconsult.net:8080/ambit2/feature/21611&feature_uris[]=http://apps.ideaconsult.net:8080/ambit2/feature/26221&feature_uris[]=http://apps.ideaconsult.net:8080/ambit2/feature/21590

More information on this topic can be found in the OpenTox API documentation.

4.3 Model Learning Example

The first building block for model learning, i.e. creating a QSAR model that later on can be used for prediction, is a dataset to train the model. Datasets can either be uploaded via a web form or via HTTP POST to the dataset service. Various formats suitable for molecular data such as SDF, CSV, SMI, MOL, and others are allowed. Alternatively, existing datasets can be used. A second building block is the descriptor/feature calculation. Before you start with learning the model, you can apply feature selection to reduce the dimensionality of your learning problem. The next building block is the actual training of the model. The model can then be used for making predictions. One can also make the model available for the ontology service and search.

For the model learning example we use the Caco-2 dataset (apps.ideaconsult.net:8080/ambit2/dataset/54) that has gastrointestinal absorption as endpoint. The endpoint feature has apps.ideaconsult.net:8080/ambit2/feature/22200 as identifying URI. For this dataset, we now calculate some additional descriptors (e.g. AcidicGroups, BasicGroups as a small example selection) that are provided in the JOELib2 package hosted as a web service at TUM. The cURL command line call to do this is the following:

> *curl -X POST -d 'dataset_uri=http://apps.ideaconsult.net:8080/ambit2/dataset/54' -d 'ALL=false' -d 'AcidicGroups=true' -d 'BasicGroups=true' http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/JOELIB2PhysChem*

The system immediately returns a URI. For long running calculations this is a task URI (e.g., opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/task/9f9ce51a-1a3e-4151-aaec-7beb8c70c38d) that can be queried for the status of the submitted calculations (e.g., in the web browser). As soon as the calculations are done you can follow the task URI link to the result URI, which is in this case a dataset URI (apps.ideaconsult.net:8080/ambit2/dataset/2628). Tasks that are completed are removed from the system after some time (at TUM this is two hours). In this example we will perform no feature selection. The next step in the workflow is the model creation. We will use the Gaussian Process Regression algorithm provided as a service at the TUM service to learn the model. The corresponding cURL command is:

> *curl -X POST -d 'dataset_uri=http://apps.ideaconsult.net:8080/ambit2/dataset/2628' -d 'prediction_feature=http://apps.ideaconsult.net:8080/ambit2/feature/22200' http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP*

The result again is a task URI. When the learning task is finished, model URI (opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/model/TUMOpenToxModel_GaussP_1) is given as a result. This model URI can then be used to predict the gastrointestinal absorption for new compounds. To investigate the performance and accuracy of the trained model you can use the validation service at ALU. An example call for a ten-fold cross-validation is:

> *curl -X POST -d algorithm_uri="http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP" -d dataset_uri="http://apps.ideaconsult.net:8080/ambit2/dataset/2628" -d prediction_feature="http://apps.ideaconsult.net:8080/ambit2/feature/22200" -d num_folds=10 -d random_seed=1 -d stratified=false http://opentox.informatik.uni-freiburg.de/validation/crossvalidation*

This call invokes the model building process – as described above – 10 times on different subsets of the dataset. Each time, the remaining (test) fold of the particular data subset is predicted by the model. A crossvalidation resource URI is returned (alike to opentox.informatik.uni-freiburg.de/validation/crossvalidation/123). This resource contains the statistics, computed from the validation service by comparing the prediction results to the actual values in the dataset.

A crossvalidation report presents the results in a nice, human readable format. It can be created with the following curl call, which returns a crossvalidation report URI that can be viewed with a web browser:

> *curl -X POST -d validation_uris="http://opentox.informatik.uni-freiburg.de/validation/crossvalidation/123" http://opentox.informatik.uni-freiburg.de/validation/report/crossvalidation*

To predict the gastrointestinal absorption for new compounds the prediction services need the same descriptors in the input dataset as the ones used for creating the model. If you know the descriptors used you can calculate them as in the descriptor calculation step. If not, you can use the super-service provided by IDEA (apps.ideaconsult.net:8080/ambit2/algorithm/superservice).

4.4 Registration of a Service at the Ontology Service

The Ontology service provides RDF storage and SPARQL search functionality for objects, defined by OpenTox services and relevant ontologies (Algorithm types, Blue Obelisk algorithms dictionary, EChA endpoints ontology and others). The query interface is a SPARQL endpoint and compliant with www.w3.org/TR/rdf-sparql-protocol/#query-bindings-http. Services are registered into the Ontology service by POSTing the URL of the service. The Ontology service retrieves RDF/XML from the URL and adds the triples into the RDF storage.

curl -X POST -d "uri=http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP" http://apps.ideaconsult.net:8080/ontology

Any OpenTox service, supporting the "application/rdf+xml" mime type can be registered via the above command. Services can be deleted by the HTTP DELETE command. This will remove all triples, associated with the URI specified.

curl -X DELETE -d "uri=http://opentox.informatik.tu-muenchen.de:8080/OpenTox-dev/algorithm/GaussP" http://apps.ideaconsult.net:8080/ontology

The current implementation of the Ontology service is based on Jena with file storage (TDB openjena.org/TDB/). Future implementations could be based on technologies, providing better performance, e.g. Jena with database backend or high performance triple storages such as Virtuoso (ods.openlinksw.com/dataspace/dav/wiki/Main/VOSRDF) or Sesame (www.openrdf.org/). Future work also may provide solutions for time limited registration of services, as well as assignment of metadata, reflecting the status of registered models (e.g. experimental or validated and ready to be used).

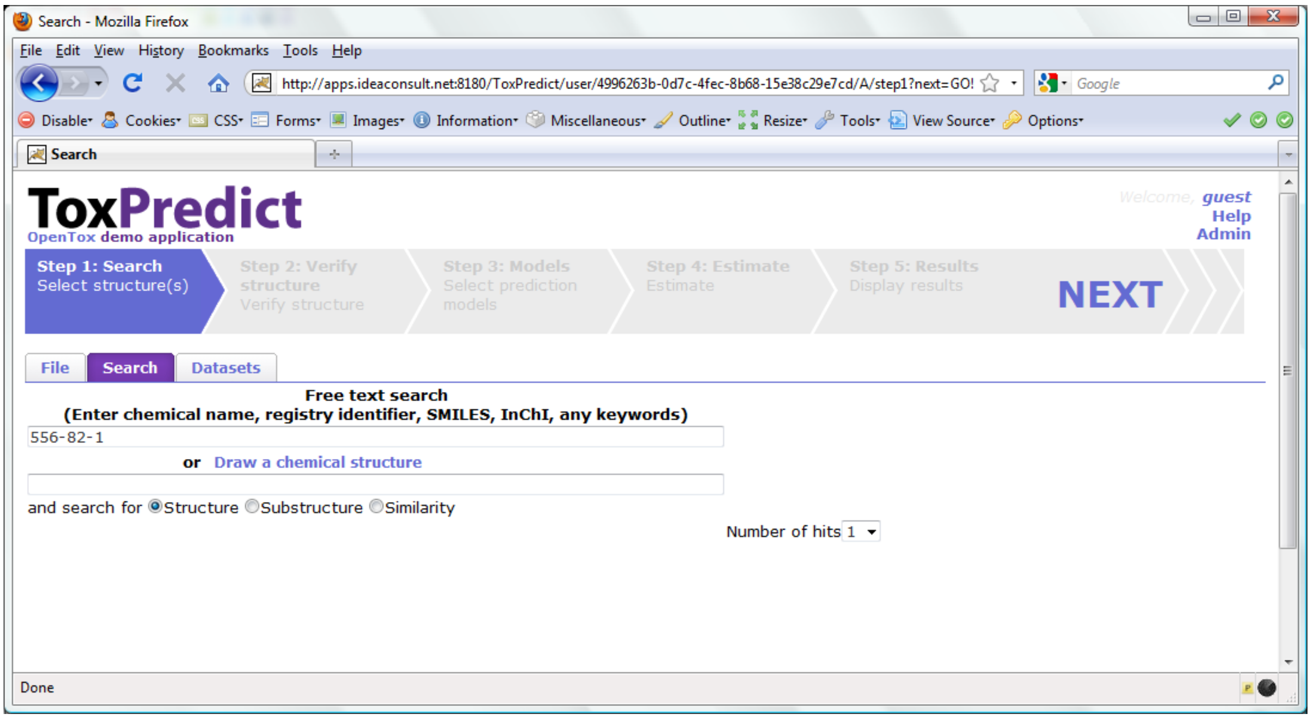
5. Example Applications

ToxPredict (toxpredict.org) estimates the chemical hazard of chemical structures. It currently relies on OpenTox API-v1.1 compliant RESTful web services. The ToxCreate (toxcreate.org) sister application allows more advanced users to train new OpenTox models. Explanatory YouTube screencasts have been available for both applications. The ToxPredict screencast (6:06 mins) is available in 360p, 480p and 720p (High Definition) resolutions at: www.youtube.com/watch?v=H7gkNBz4Lwo. The ToxCreate screencast (5:52 mins) is available in 360p, 480p and 720p (High Definition) resolutions at: www.youtube.com/watch?v=JRIl_XbibMA.

5.1 ToxPredict

To estimate the hazard of a chemical structure, ToxPredict users can either search the OpenTox prototype database, which currently includes quality labeled data for thousands of REACH-relevant chemicals, grouped in datasets, or can upload their own chemical structure data. ToxPredict provides access to a number of ready-to-use models, addressing various endpoints.

ToxPredict's Graphical User Interface (GUI) includes three main areas: a) a navigation bar at the top of the screen, which enables users to find their way through the application workflow; b) the navigation bar is followed by a short contextual help section, providing specific details for the current step being executed; and c) an input/output section for specifying queries and displaying results situated at the bottom of the screen.

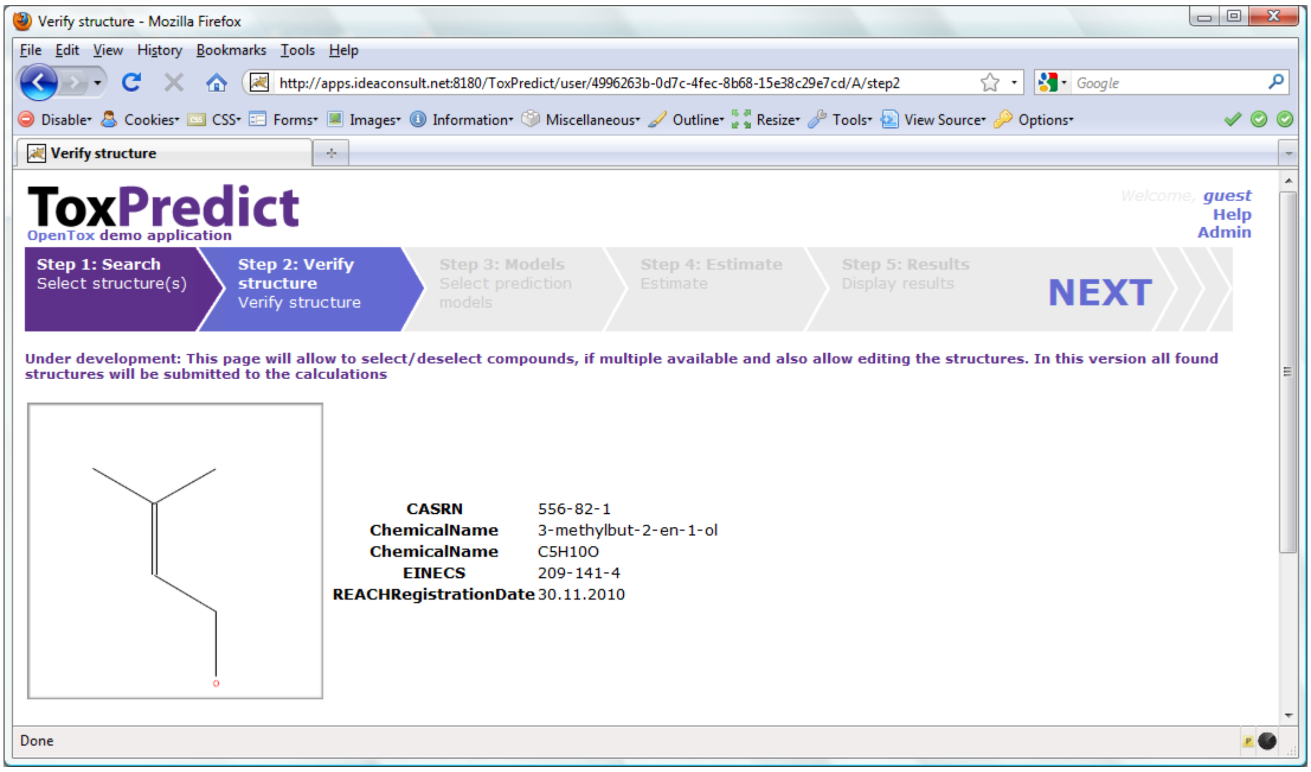


Free text search
(Enter chemical name, registry identifier, SMILES, InChI, any keywords)
556-82-1
or Draw a chemical structure
and search for [✓ Structure] Substructure Similarity
Number of hits 1

Figure 2 ToxPredict GUI: Step 1 – Enter/select a chemical compound

In the first step of the ToxPredict workflow shown in Figure 2, you are invited to select the structure(s) for which you would like to apply some OpenTox model(s). Once you have entered your query string, chemical structure drawing or file name for upload, pressing the NEXT button will continue the ToxPredict workflow to the next step.

In the second step shown in Figure 3, you can browse the list of chemicals, which have been found to correspond to the query defined in the first step. The structure of the chemicals can be seen on the left hand side of the screen and some further metadata such as CAS registry number, EINECS number, IUPAC name, synonyms and quality label are shown on the right. In case that you are satisfied with this list you can proceed to the following step, otherwise you could go back to the first step of the workflow and define a different query.


Verify structure - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://apps.ideaconsult.net:8180/ToxPredict/user/4996263b-0d7c-4fec-8b68-15e38c29e7cd/A/step2
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Verify structure
ToxPredict
OpenTox demo application
Welcome, guest
Help
Admin
Step 1: Search
Select structure(s)
Step 2: Verify structure
Verify structure
Step 3: Models
Select prediction models
Step 4: Estimate
Estimate
Step 5: Results
Display results
NEXT
Done

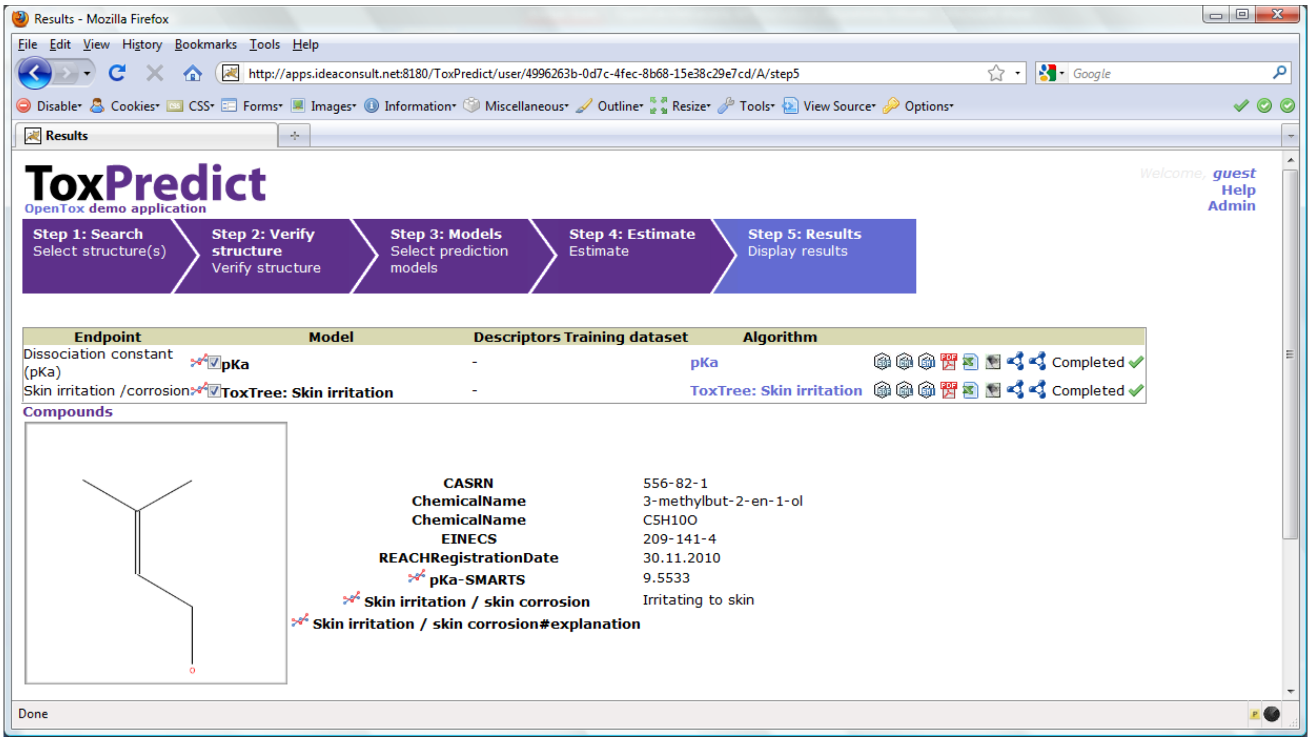
Under development: This page will allow to select/deselect compounds, if multiple available and also allow editing the structures. In this version all found structures will be submitted to the calculations

CASRN	556-82-1
ChemicalName	3-methylbut-2-en-1-ol
ChemicalName	C5H10O
EINECS	209-141-4
REACHRegistrationDate	30.11.2010

Figure 3 ToxPredict GUI: Step 2 Display selected/found structures

The following step lists the available OpenTox models that can be selected for estimating various properties for each of the query chemicals along with the corresponding endpoints. It is also possible to select all available models at once.

In the next step the selected models are applied to all query chemicals and you can follow the processing progress, which is refreshed automatically every 30 seconds. After the processing has been completed, you can go to the last step of the ToxPredict workflow to get the detailed results for the selected models and compounds, displayed on a web page and also available for download as a SDF file, which is convenient for further automatic processing. An example results page is shown in Figure 4.



Results - Mozilla Firefox

http://apps.ideaconsult.net:8180/ToxPredict/user/4996263b-0d7c-4fec-8b68-15e38c29e7cd/A/step5

ToxPredict
OpenTox demo application

Welcome, guest
Help
Admin

Step 1: Search Select structure(s) | **Step 2: Verify structure** Verify structure | **Step 3: Models** Select prediction models | **Step 4: Estimate** Estimate | **Step 5: Results** Display 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
ChemicalName	C5H10O
EINECS	209-141-4
REACHRegistrationDate	30.11.2010
pKa-SMARTS	9.5533
Skin irritation / skin corrosion	Irritating to skin
Skin irritation / skin corrosion#explanation	

Done

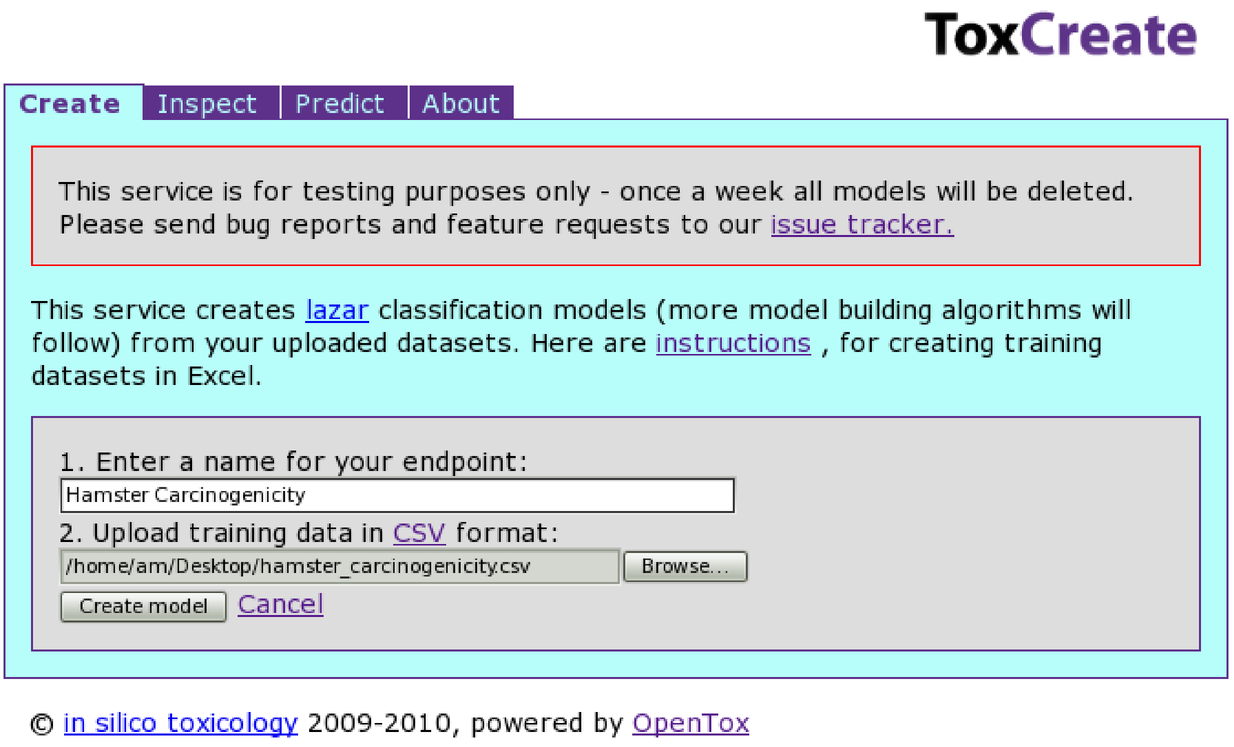
Figure 4 ToxPredict GUI: Final Step – Display toxicity data and prediction results

5.2 ToxCreate

The first version of ToxCreate (toxcreate.org) creates lazar classification models from user uploaded datasets; additional model building algorithms are planned to follow in subsequent versions.

The first step involves the input of data for model building as illustrated in Figure 5. The input file should contain two columns, separated by a comma. The first column is reserved for the chemical structure in SMILES format, while in the second column the user should specify the activity classification (1: active, 0: inactive). An example input file is available for download at toxcreate.org/hamster_carcinogenicity.csv. Input files can also be created with MS Excel: Create a sheet with two columns and export them as a CSV (comma delimited) file with the "Save As" option from the menu, selecting the CSV format.

After submitting a dataset to ToxCreate, an endpoint name, e.g., EPA Fathead Minnow (EPAFHM) acute toxicity, should be assigned before proceeding with the file upload and creating the model. This can take a while, depending on the dataset size and the current load on the service. The Inspect page is refreshed every 15 seconds to update the model status. Once the model building is completed, you can go to the Predict page and apply the newly created model to predict the EPAFHM endpoint for some chemicals, identified by structure diagram drawing, name, InChI, CAS, SMILES or one of the further possibilities offered. One or more prediction models have to be selected. Example results are shown in Figure 6.



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:
Hamster Carcinogenicity
2. Upload training data in CSV format:
/home/am/Desktop/hamster_carcinogenicity.csv Browse...
Create model Cancel

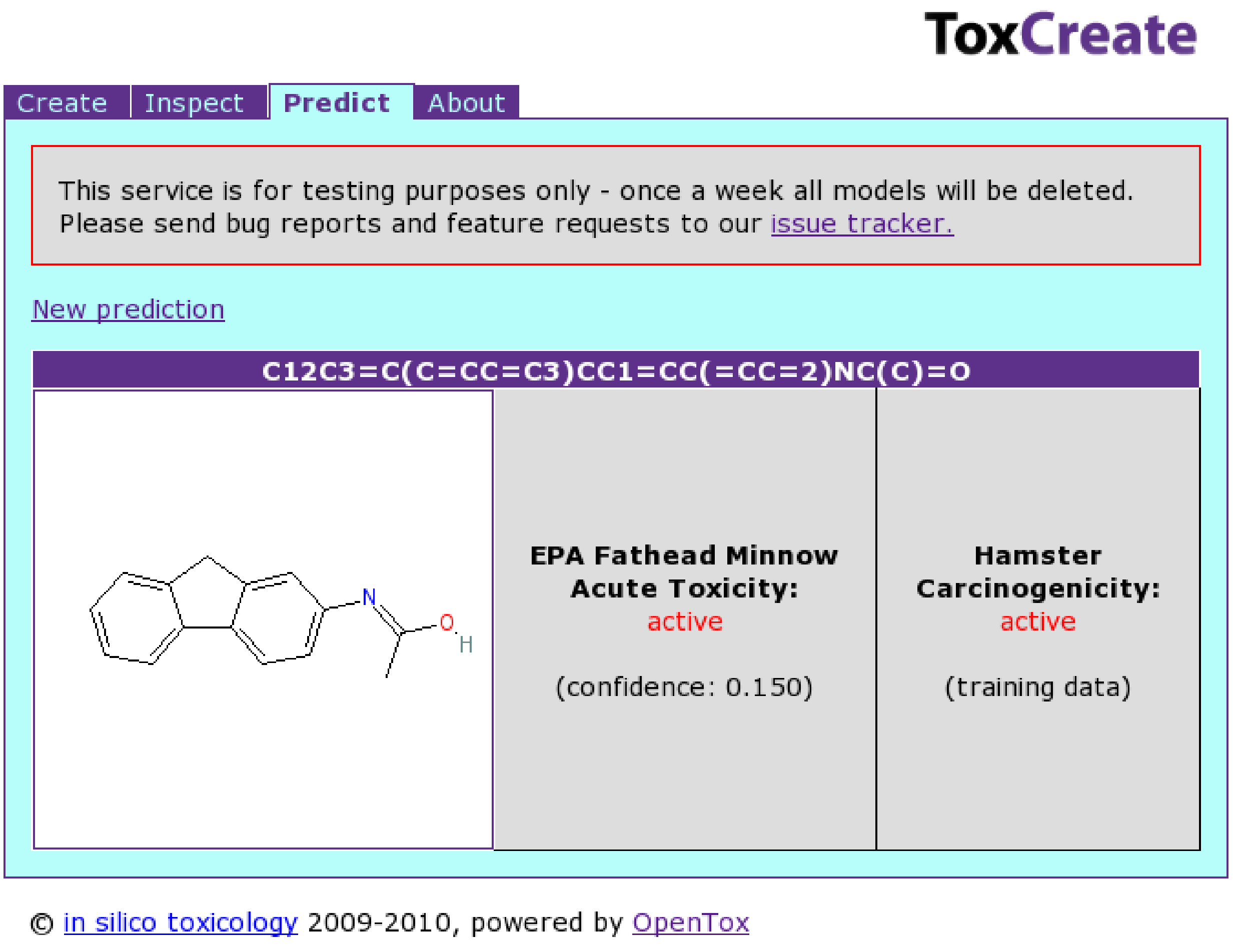
© in silico toxicology 2009-2010, powered by OpenTox

Figure 5 ToxCreate GUI:Initial Step – Input data

5.3 Connecting OpenTox Services with Taverna

Taverna (www.taverna.org.uk) provides a workflow system to access and combine web services. It provides a user interface which can be used to generate arbitrary workflows from combinations of single web services. The web services are combined by importing single web services and connecting their inputs and outputs using a point-and-click GUI. The GUI makes it a candidate application for users with little programming knowledge to create their own workflows.

To show that the OpenTox API and web services can be combined with the Taverna workflow system, some basic workflows were developed, which can be combined to create more complex workflows. These workflows have been made available through the myexperiment web site at www.myexperiment.org/groups/247.html. They are BeanShell scripts (www.beanshell.org) which use basic Java functionality to access the OpenTox web services via HTTP.



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:	Hamster Carcinogenicity:
	active (confidence: 0.150)	active (training data)

© in silico toxicology 2009-2010, powered by OpenTox

Figure 6 ToxCreate GUI: Display model prediction results

Using the myexperiment platform, the workflows can be shared and discussed with others. They are also available for download and can then be imported into Taverna using the import of a "Nested workflow" functionality. The nested workflow has input and output ports. Each input can be connected to an output and vice-versa, enabling complex workflows to be generated step by step. An example complex workflow is uploaded to myexperiment and available at www.myexperiment.org/workflows/1119.html; screenshots of the workflow are provided in Appendix B. When running the workflow, the user is asked for a test set, a training set and a target attribute. In the example workflow, the datasets are uploaded to the IDEA ambit data web services before a model is learned using a TUM web service. Finally, the model is used to predict the target attribute of the test set and the URL of the resulting test set with the target attribute is returned.

6. Conclusions

This report provides an overview on the status and capabilities of the OpenTox prototype system, designed and developed in the first two years of the project. The OpenTox prototype relies on a distributed REST-based web service architecture with services distributed over multiple sites and organisations. The ToxPredict and ToxCreate applications and the Taverna workflow integration examples show the flexibility of the system. Future versions (starting with API version 1.2 which is due for release in late 2010) will include capabilities for Authentication and Authorisation and additional algorithms for calculating the applicability domain of a prediction model.

A key decision for the OpenTox prototype was the adoption of the REST architectural style, because it is suitable for achieving three important goals: independent deployment of components, ease of standardised communication between components and generality of interfaces. These advantages will enable the development and integration of additional algorithms in the future, which may be offered by a variety of third-party developers in the community. Ongoing maintenance and addition of novel predictive algorithms relevant to predictive toxicology will contribute to the long-term sustainability of OpenTox in generating valuable resources for the user scientific community.

Continuing effort will be carried out by OpenTox developers to meet current academic and industry challenges regarding interoperability of software components and integration of data, algorithm and model services within the context of tested Use Cases. The experience we have gained during this work should help speed up the development process towards this direction. The approach to interoperability and standards lays a solid foundation to extend application development within the broader developer community to establish computing capabilities that are sorely missing in the field of predictive toxicology today, and which are holding back advances in both R&D and the application of R&D project outcomes to meet industry and regulatory needs.

7. References

[BUC10] Buchwald, F, Girschick, T, Frank, E, and Kramer, S (2010). Fast Conditional Density Estimation for Quantitative Structure-Activity Relationships In: Proceedings of the 24th AAAI Conference on Artificial Intelligence, pp. 1268-1273, AAAI Press.

[FIE00] Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine (2000)

[HAR10] Hardy, B., Douglas, N., Helma, C., Rautenberg, M., Jeliazkova, N., Jeliazkov, V.,Nikolova, I., Benigni, R., Tcheremenskaia, O., Kramer, S., Girschick, T., Buchwald,F., Wicker, J., Karwath, A., Gütlein, M., Maunz, A., Sarimveis, H., Melagraki, G.,Afantitis, A., Sopasakis, P., Gallagher, D., Poroikov, V., Filimonov, D., Zakharov, A., Lagunin, A., Gloriozova, T., Novikov, S., Skvortsova, N., Druzhilovsky, D., Chawla, S., Ghosh, I., Ray, S., Patel, H., Escher, S.: Collaborative Development of Predictive Toxicology Applications. Journal of Cheminformatics, 2:7 doi:10.1186/1758-2946-2-7, Available in Open Access at www.jcheminf.com/content/2/1/7 (2010)

[HEL06] C. Helma. Lazy structure-activity relationships (lazar) for the prediction of rodent carcinogenicity and Salmonella mutagenicity. *Molecular Diversity*, 10:147-158, (2006)

[MAU10] Maunz, A., Helma, C., Kramer, S.: Efficient Mining for Structurally Diverse Subgraph Patterns in Large Molecular Databases. Machine Learning, in press (2010)

[PAT08] Patlewicz, G., Jeliazkova, N., Safford, R.J., Worth, A.P., Aleksiev, B.: An Evaluation of the Implementation of the Cramer Classification Scheme in the Toxtree Software. SAR & QSAR in Environmental Research 19(5-6) (2008) 495-524

Appendix A – Example Java Resource

```
package opentox.algorithm.learning;
/**
* Import statements left out for brevity of this document. The complete TUM source code is
* available at:
* http://opentox.informatik.tu-muenchen.de/trac/TUMOpenTox/browser/trunk/src/opentox
* [...] means the code/text has been abbreviated
*/

/**
 * M5P regression algorithm from the WEKA workbench (http://www.cs.waikato.ac.nz/~ml/weka/)
 * wrapped as OpenTox REST web service.
 * […]
 * @author girschic, buchwald
 */
public class M5PResource extends AbstractLearningAlgorithm{

        /**
         * Constructor
         * @param context
         * @param request
         * @param response
         */
        public M5PResource(Context context, Request request, Response response) {
                /*
                 * the buildRDF() and setDefaultParameters method are
                 * called in the super constructor (AbstractAlgorithm)
                 */
                super(context, request, response);
                // […]
        this.setModelID(getRequest().getRootRef().getIdentifier()+"/model/TUMOpenToxModel_M5P_"+
idCount);

                this.getVariants().add(new Variant(MediaType.TEXT_HTML));
                this.getVariants().add(new Variant(MediaType.TEXT_XML));
                this.getVariants().add(new Variant(MediaType.APPLICATION_RDF_XML));
                this.getVariants().add(new Variant(MediaType.APPLICATION_RDF_TURTLE));
        }


        @Override
        public void acceptRepresentation(Representation entity) {
                String model_uri = "";
                String uri = "";
                boolean paramsOK = true;
                String pred_feat = "";
                String datasetservice = null;

        try{
                if (entity != null) {
                        if (MediaType.APPLICATION_WWW_FORM.equals(entity.getMediaType(), true)) {
                                Form form = new Form(entity);
                                Map<String,String> m = form.getValuesMap();
                                uri = form.getValues("dataset_uri");
                                pred_feat = form.getValues("prediction_feature");
                                datasetservice = form.getValues("dataset_service");



        /*
         * […] parameter and further input checking left out
         * In the callable object the actual calculations take place. As model creation can take long,
         * it is realized as an asynchronous task and an OpenTox task URI is returned immediately.
         */

        if(paramsOK == true){
                URI fileURI;
                try {
                                fileURI = new URI(uri);
                                URI pred = new URI(pred_feat);
                                CallableM5P callable = new CallableM5P(fileURI, pred, getRequest());
                                callable.setModelURI(getModelID());
                                callable.setParams(params);
                                callable.setDatasetService(datasetservice);

                        Reference taskref = ((OpenToxApplicationTUM)getApplication()).addTask(callable,
getRequest().getRootRef());
                        /*
                         * Also set Location: header
                         */
                        getResponse().setLocationRef(taskref);


                         getResponse().setEntity(new StringRepresentation(taskref.getIdentifier(),
MediaType.TEXT_URI_LIST));
                                getResponse().setStatus(Status.SUCCESS_ACCEPTED);
                        } catch (URISyntaxException e) {
                                e.printStackTrace();
                        }
        }
```

Appendix B – Taverna OpenTox Workflow

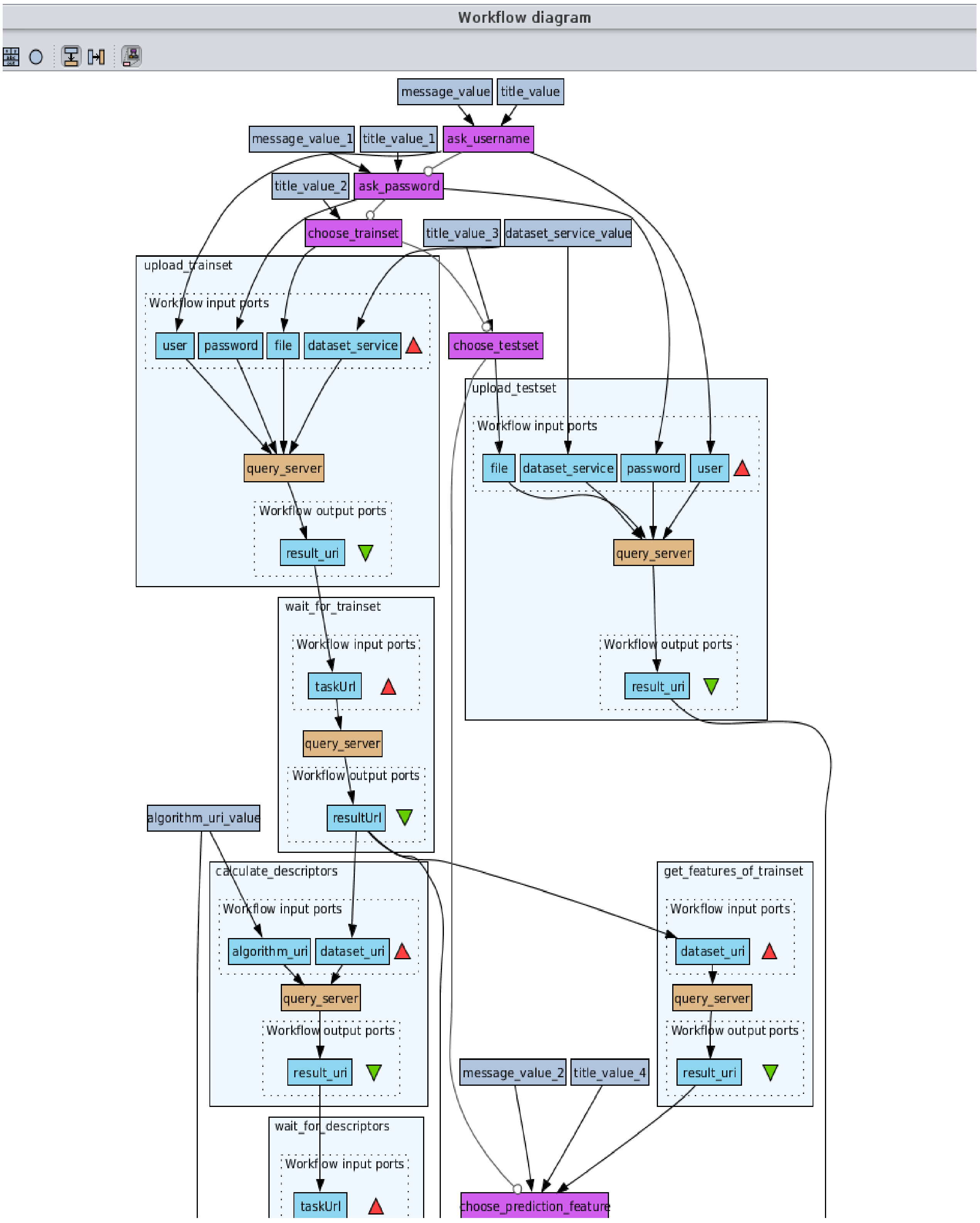


Figure B1 Part one of a screenshot of the Taverna OpenTox example workflow. Available for download at www.myexperiment.org/workflows/1119.html .

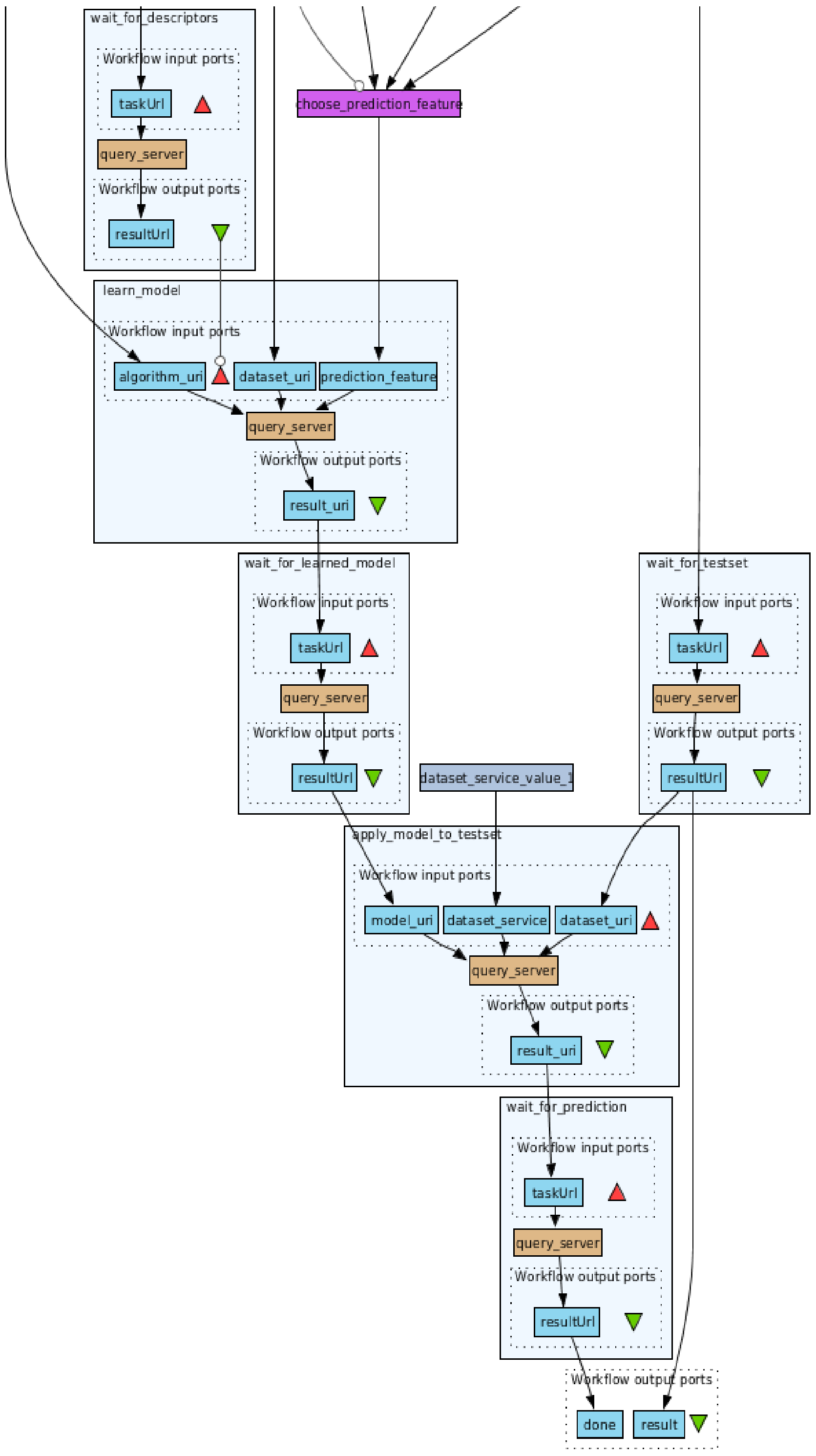


Figure B2 Part two of a screenshot of the Taverna OpenTox example workflow.