HelmholtzZentrum münchen
Deutsches Forschungszentrum für Gesundheit und Umwelt

Marie Curie Initial Training Network
Environmental Chemoinformatics (ECO)

Final Report of STF

Environmental Probabilistic Risk Assessment

July 31, 2012

Early stage researcher:
Pantelis Sopasakis

Project supervisor:
Igor Tetko

Research Institution:
HelmholtzZentrum München

Final Report

Pantelis Sopasakis

CONTENTS

Abstract

In this report, the main goal of my work with the Helmholtz Research Centre in Munich is summarised. I developed a computational platform to serve as a statistical inference tool which combines statistical methods, QSAR models and is based on modern Web technologies. The developed web tool will allow risk assessors to use the information provided by QSAR models and experimental measurements to assess the possible environmental damage provoked by the emission of a chemical to the ecosystem.

Index Terms

Environmental Cheminformatics, Statistical Inference, Web Services

Pantelis Sopasakis is with the Helmholtz Zentrum Muenchen. Email: chvng@mail.ntua.gr.

I. Project Overview

Project Summary	
Project Name	Environmental Probabilistic Risk Assessment
Supervisor	Dr. Igor Tetko
Description	Use of QSAR/QSPR models to assess the environmental risk related to the emission of chemical compounds to the environment
Project Period	16/04/2012 - 16/07/2012
Report Date	July 31, 2012

This project was carried out in close collaboration with partners from the FP7-funded EU research project *CADASTER* [1]. The research leading to the results outlined hereinafter has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013] under Grant Agreement n. 238701.

II. Introduction

A. Motivation

THE advent of *statistical inference* as a rigorous and well-established analysis methodology could not have gone unnoticed by environmental engineering [2]. Statistical Inference and Probabilistic Risk Assessment (PRA) have been extensively employed in complex technological units – such as nuclear plants [3] or socio-technical systems [4] – to assess the risk of a possible action. In environmental cheminformatics the key question we need to answer is whether the emission of a certain chemical to the ecosystem can cause environmental damage (in terms of toxic effects to the flora and fauna of the ecosystem or bioaccumulation which in turn may lead to long-term toxicity).

QSAR/QSPR models are extensively used to predict physicochemical, chemical and biological properties of chemical substances for which there is lack of experimental measurements. PRA assumes and requires that all predictive models involved in the overall workflow provide an estimation of the uncertainty of the predictions they output; these predictions are in fact probability distributions. Available experimental values along with QSAR predictions are fed to an *exposure* and/or *effect* model which estimates the environmental fate of the respective chemical compound.

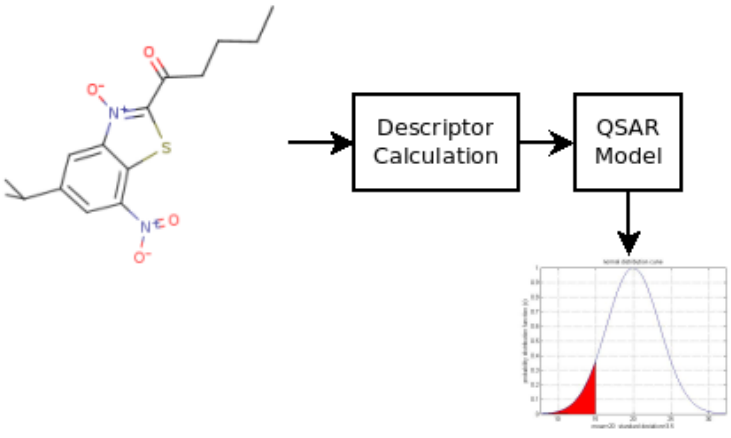

Fig. 1. In PRA, predictive models are assumed to output not just a single deterministic prediction, but a probability distribution function describing the uncertainty of the prediction.

The sources of uncertainty to be considered in a probabilistic risk assessment study are:

1) Uncertainty of the output of QSAR models (modelling error)
2) Uncertainty induced by the use of uncertain experimental measurements in QSAR models

3) Uncertainty of experimental measurements (provided to the exposure and effect assessment model)
4) Mismatch between the modelled ecosystem and the real one including time-dependent fluctuations of the local population dynamics and chemical state (pH of water, concentration of ions etc) that may occur either naturally or due to human activities.

These sources of uncertainty are difficult to track as extensive experimental work needs to be done. Nonetheless, to a great extent most of these effects can be taken into account. This does not lead to a complete PRA schema but rather to the formulation of a statistical inference problem.

Statistical Inference or *Statistical Induction* is the process of arriving at conclusions whose validity is statistically controlled from a set of available data that is subject to uncertainty by sampling from a general population. Propositions about a population are made using observations and data drawn from the general distribution [5], [6]. The outcome of such an analysis is a statistical proposition which can be formulated as a simple estimate (expected value), a confidence interval for a given level of confidence, the rejection or acceptance of a hypothesis and the accompanying P-value, or a probability distribution function.

In our study, a statistical inference schema was adopted which outputs a probability distribution function constructed using Monte-Carlo simulations. The target properties quantify the accumulation potential of the substance in the ecosystem as we will elaborate on in the following sections.

B. Overview

Various assessment models for the *exposure* and/or *effect* have been developed within the CADASTER project [1] based on Simplebox [7]. Each model has a chemically specified domain of applicability; in particular there have been developed models for Triazoles/Benzotriazoles, BDEs (Brominated Diphenylethers), PBDEs (Polybrominated Diphenylethers), PFCs (Perfluorinated and Perfluoroalkylated Compounds) and Substituted Musks/Fragrances. The exposure models require certain physicochemical input parameters such as their solubility in water, melting point, vapour pressure, bulk degradation rate constant in standard soil and a lot more. These are either provided as (known) experimental values or are predicted using relevant QSAR/QSPR models. In the latter case, the prediction obtained by the model brings about an intrinsic uncertainty of the QSAR model which should be taken into account in the risk assessment study. Of course, experimental measurements come always along with an uncertainty estimate (e.g. standard deviation)

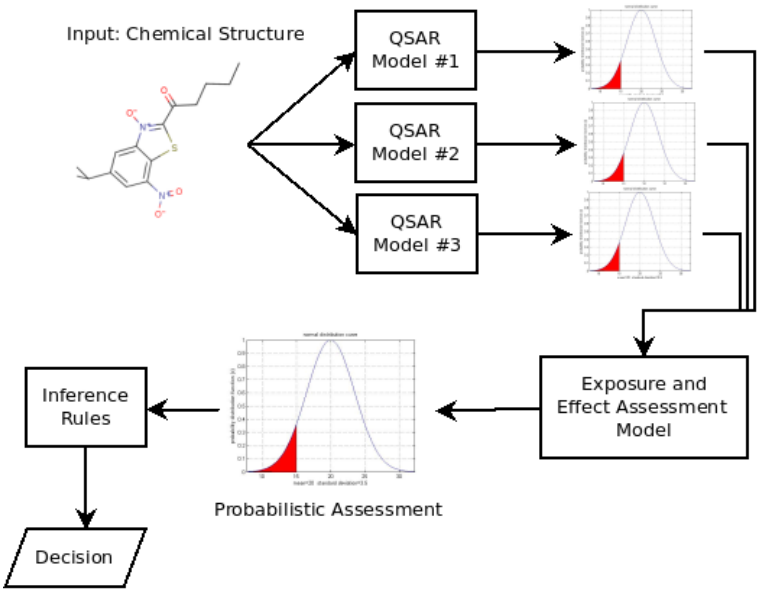

Fig. 2. Overall workflow of PRA using a set of QSAR models and an exposure and effect assessment model.

III. Implementation - Design Considerations

The above schema is being implemented in the CADASTER project allowing end-users to perform PRA analyses on selected chemical compounds. Hereinafter we shall elaborate on various aspects of the implementation and the

underlying computational concept.

A. Prediction Uncertainty

Every prediction model brings about an uncertainty when providing a prediction for an unknown input instance which quantifies the reliability of this prediction [8]. A plain prediction output is devoid of any meaning without an estimation of its uncertainty. This gives rise to the notion of the *applicability domain* assessment of a QSAR model [9], [10], [11]. The predictions made for compounds that lie inside the domain of applicability of a given model are expected to be more reliable and have the accuracy of predictions that one makes over the training instances of the model, while for the ones outside the domain of applicability no assumption can be safely posed.

The uncertainty of a predictive model – especially in terms of probability distributions – is in general an open issue for nonlinear models. To formalise the problem, consider a predictive model described by the nonlinear function $f : \mathbb{R}^n \to \mathbb{R}$ and let x be an input vector and y_x the actual target value corresponding to x. Then, we define the error of the model output at x as $e(x) = y_x - f(x)$. In general, the prediction error is not known beforehand. Let $d_f : \mathbb{R}^n \to \mathbb{R}_+$ be a continuous function and define the function:

$$E(d) = \mathcal{E}\left\{e = y_{x_i} - f(x_i), d_f(x_i) = d\right\}_{i \in \mathbb{I}}, \tag{1}$$

where $\mathcal{E}$ stands for the expected value operator. If the following implication holds:

$$d_f(x_1) \leq d_f(x_2) \Rightarrow E(d_f(x_1)) \leq E(d_f(x_2)), \tag{2}$$

then d_f is said to be a *distance to model metric* with respect to the model f.

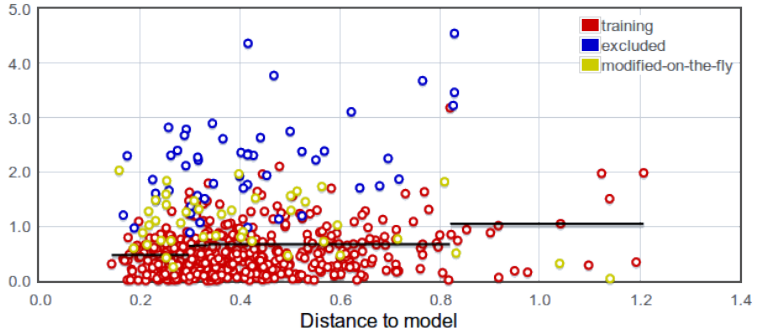

Fig. 3. A distance-to-model versus modelling error diagram. The distance-to-model function in this example is piecewise constant and nondecreasing. This diagram was output from http://qspr-thesaurus.eu/

Condition (2) however is not one that can be directly tested in practice as the error of every prediction is not known. As a result one should restrict to the sets of data points for which experimental measurements are available to derive relevant conclusions. But, again, the expected value cannot be estimated as the set $\{e = y_{x_i} - f(x_i) | d_f(x_i) = d; i \in \mathbb{I}\}$ is bound to be empty with probability 1. For this reason we restrict our attention to the special class of piecewise-constant and nondecreasing functions and we require that d_f is such. In that case the error is estimated over an interval and its derivation is described in [12].

The notion of a *domain of applicability* for a QSAR model in this context is meant in two complementary manners. First, a compound is considered to be inside the DoA of a given QSAR model if the training set of the latter has *enough* training vectors "close" to it. This proximity is quantified by the aforementioned metric referred to as a *distance-to-model function* [9]. There is however a second approach to the domain of applicability; that of chemical similarity which is not always cast by the first approach (recall the babies-and-storks paradox [13]). This calls for our increased attention since a predictive model for triazole derivatives – trained using a set of such compounds – should not be used for other chemicals even if some might exhibit low distance-to-model values.

Another evaluation of the Applicability Domain comes from expert knowledge which is based on external (to the model) information about chemical compounds. To some extent this knowledge plays the role of a *prior* information in the Bayesian analysis.

B. Monte Carlo Simulations

Let $x \in \mathbb{R}^n$ be a random variable that follows the distribution $x \sim F$ and $T : \mathbb{R}^n \to \mathbb{R}^m$ a Borel measurable mapping. Then $y = T(x)$ is also a random variable whose probability distribution function (pdf) is dictated by both F and T. It can be easily proven for example that if $x \sim \mathcal{N}(\mu, \Sigma)$ and T is an affine transformation $T = a + bx$, then $y \sim \mathcal{N}(a + b\mu, b\Sigma b')$. However, finding the pdf of y is far from trivial in most cases when T is an arbitrary nonlinear function (even if F is the normal distribution); this is why we need to resort to approximations using the well known Monte-Carlo approach [14].

An exposure and effect assessment model maps a set of pdfs – which accrue from QSAR models – to pdfs describing the exposure in terms of long-range transport (LRTP), persistency half-time and predicted environmental concentration (PEC).

From an implementation point of view, various exposure and effect assessment models have been developed and are available from the Excel-based software Simplebox. These can be accessed using R scripts which undertake the Monte-Carlo simulations given a set of input distributions. For that, the library RDCOMClient library for R was used [15]. This computational process is wrapped as a web service implemented in Java which exposes its functionality to the clients by a well defined *API*.

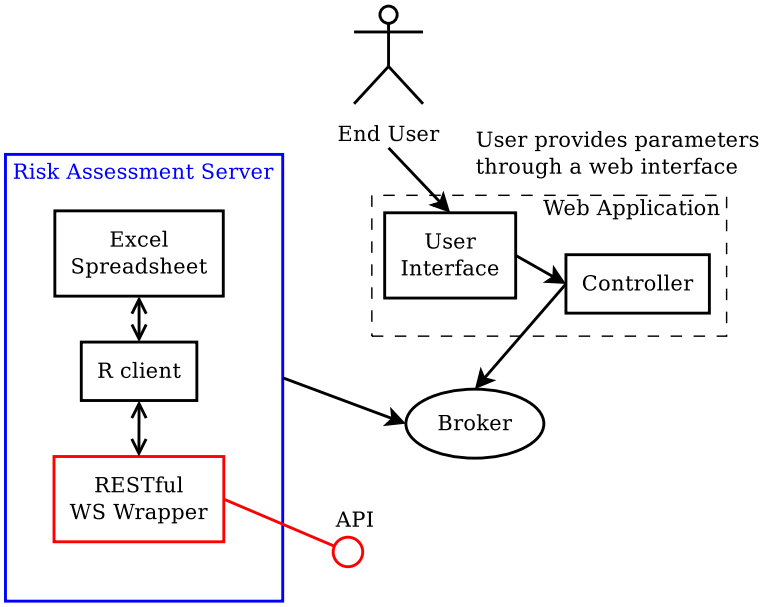

Fig. 4. Workflow of the web application being implemented. The broker distributes passively the tasks among multiple worker servers.

C. User Interface

A web-based user interface is provided to allow users to access the aforementioned computational procedure. The input expected from the user is:

1) A chemical compound (which can be either provided as SMILES, drawn or looked up for in the CADASTER database).
2) The chemical class into which the selected compound lies (BDEs, triazoles, PFCs or fragrances).
3) Experimental physicochemical parameters if available.
4) A predictive QSAR/QSPR model used to calculate the physicochemical properties for which there are no available measurements.
5) The emission scenario: where is the compound expected to be emitted (soil, water, air).

The expected output comprises of the pdfs for LRTP, PEC and persistency.

D. Application-Programming Interface

An Application-Programming Interface is a specification that is implemented by the different components of some software or of a network of machines that orchestrates their communication. A Web API defined the exact format

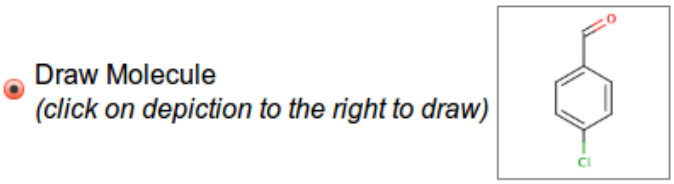

Fig. 5. Screenshot from the web user interface used to submit compounds.

of the HTTP requests (Header and Body) and also provides a schema definition for the expected response. The REST (REpresentational State Transfer) architecture was chosen for this implementation for a number of reasons that will be elucidated in the next section.

An API was specified for the worker servers that undertake the computation of probability distributions for all environmental parameters. The API is based on REST principles to maximise transparency and interoperability as it does not depend on a fixed client implementation. This makes the implemented service reusable and extensible.

Behind the scenes, Simplebox is used as an exposure and effect assessment model. Certain modifications had to be done thereupon to cast it as a runnable script callable from Java. The final output from this script is converted into a JSON file which is returned to the client. JSON files are ideal when working in a JavaScript environment as they can be parsed easily and effectively.

A web interface is currently being developed within the CADASTER project and a preliminary version of it can be found at http://qspr-thesaurus.eu/.

A preliminary version of these results were presented at the Summer School of the ECO project in Verona, Italy, 10-15 June, 2012 [16].

IV. Implementation - API

A. REST

Representational State Transfer, abbreviated as REST, is a software architecture style but does not define neither a protocol nor a framework. It is more of a concept for developing web-based (possibly distributed) applications. Its main characteristic is that it is resource-oriented. Every object or entity (resource) is named and addressable by means of a Uniform Resource Identifier which may not necessarily correspond to a web address. It is rather an identifier in the form of a web address. The principles underlying REST are the same with HTTP; the four basic and well-established methods GET, POST, PUT and DELETE are the four cornerstones of REST. The fact that all resources in REST are addressable is the main asset of this design approach in contrast to other architectures such as SOAP. On the other hand, *equal* resources (even resources with all representations perfectly identical) are not guaranteed to have equal identifiers over a distributed network of services. There are certain solutions to this kind of quirks but they require proper infrastructures (e.g. a central registry of resource identifiers, or client-side comparison of resources). At the bottom line of this short introduction, it should be stressed that REST is not a catholicon; it primarily depends on the problem definition and its requirements. In all cases there are certain principles to be taken into account. There are: Interoperability, Security, Performance, Robustness, Scalability and of course the CAP trilogy: Consistency, Partition-tolerance and Availability [17]. REST is not consistent but, if necessary, can be *eventually consistent*.

The REST API defines the way the end-user accesses the functionality provided by SBWS. The HTTP methods GET, POST and DELETE are used when invoking the web services. GET is used in two cases. One is when a list of URIs should be returned in the specified MIME (usually text/uri-list). The second is for the retrieval of a representation of a particular resource in a given media-type. POST, when applied, creates a new resource and returns it URI or a representation of the resource itself in the requested MIME. DELETE is used to delete a resource from the server.

B. HTTP Status Codes

The following HTTP status codes are employed in the application:

HTTP Status Codes	
Status Code	**Explanation**
200	Successful retrieval of the resource
201	An asynchronous task has finished execution. Another task undertakes the rest.
202	Running task
400	Bad Request
404	Resource not found Misspelled URI
500	The server encountered an internal problem
502	Some remote server encountered an error
503	The server is overloaded at the moment

All error codes (greater than or equal to 400), are always accompanied by an error report that helps the debugging. HTTP status codes have well established interpretation and promote a machine- readable setting in which the server and the client speak the same language. More information regarding the meaning of HTTP headers is provided within the RFC2616 specification which can be found at www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

C. Media Types

In this version two media types are supported. One is `text/uri-list` which is used only for lists of URIs. The second, which is supported for all resources, is `application/json`. Each JSON entity defines a type through the `object_type` parameter to facilitate parsing. Additionally, all components in SBWS bear a metadata field called metadata which points to a MetaData object The desired MIME must be specified with each HTTP request using the `Accept` header.

D. Components and Web Services

1) Metadata: The Dublin Core ontology is used to describe metadata and to serialise them in JSON. More MIME types might be supported in the future as well such as RDF/XML. We use the following properties from the DC ontology:

Metadata Specification		
Ontological Property	**Explanation**	**JSON Property**
dc:description	An account of the content of the resource. Description may include but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content.	description
rdfs:comment	rdfs:comment is used to provide a human-readable description of a resource. Users can specify a set of such comments on a resource of interest.	comment
dc:rights	Copyright notes for the underlying resource.	rights
dc:identifiers	The identifier (ID) of the described entity. An identifier is an unambiguous reference to the resource within a given context. Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Examples of formal identification systems include the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL), the Digital Object Identifier (DOI) and the International Standard Book Number (ISBN).	identifier
rdfs:seeAlso	The property rdfs:seeAlso specifies a resource that might provide additional information about the subject resource. This property may be specialised using rdfs:subPropertyOf to more precisely indicate the nature of the information the object resource has about the subject resource. The object and the subject resources are constrained only to be instances of the class rdfs:Resource.	seeAlso
dc:title	The name given to the resource. Typically, a Title will be a name by which the resource is formally known.	title
dc:subject	The topic of the content of the resource. Typically, a Subject will be expressed as keywords or key phrases or classification codes that describe the topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.	subject

Example of metadata object serialised as JSON:

```
{
  "object_type":"metadata",
  "title":[
    "My Resource"
  ],
  "subject":[
    "Example JSON"
  ],
  "description":[
    "Description of the entity goes here"
  ],
  "comments":[
    "Comment 1",
    "Comment 2"
  ]
}
```

As in this example, all fields in a JSON representation of a metadata component are of type array following the DC and RDFS specification and definition of the properties we use in the MetaData object.

2) Parameter: A Parameter defines a specific instance of a feature-to-value mapping along with the necessary metadata to communicate its logical width to the client. Every parameter defines a name or set of names that is provided using the dc:title property. A `scope` is included to differentiate between `optional` and `mandatory` parameters.

In the following table the admissible parameters for this class are specified:

Parameter: JSON Specification		
Field	**Explanation**	**Admissible Values**
`metadata`	An instance of MetaData	`MetaData` objects
`scope`	The scope of the parameter	Mandatory, Optional
`value`	The value of the parameter	Double or String
`units`	Units of the parameter	String
`valueType`	Type as which `value` should be cast.	A Java Type (Double, Integer, Float, String, URI)
`name`	The name of the parameter (can also be specified within the metadata as title).	String

Here is an example JSON for a parameter:

```
{
  "object_type":"Parameter",
  "name":"N",
  "scope":"OPTIONAL",
  "value":"100.0",
  "valueType":"Float",
  "units":"unitless",
  "metadata":{
    "object_type":"MetaData",
    "title":"N",
    "identifier":[
      "4b6eea6a-d504-4180-973b-e3f695b8ad0f"
    ]
  }
}
```

3) Algorithm: Algorithms provide access to the main functionality of Simplebox that runs behind the scenes. A list of all available algorithms is returned at /algorithm specifying the MIME type text/uri-list. Algorithms can be queried using the URL parameter ?type=type where type is any keyword.

Individual algorithms follow the URI template /algorithm/id. These resources upon OPTIONS define the way they should be invoked so that a client application can be created quite automatically and can synchronise with eminent changes in the API. Upon GET, the algorithm resource returns metadata information such as the required (optional and mandatory) parameters it requires and their meaning as well as a set of DC (Dublin Core) metadata properties.

Algorithm: WS Specification					
Description	**Method**	**URI**	**Parameters**	**Result**	**Status**
Get URIs of all algorithms	GET	/algorithm	type	List of all implemented algorithms	200, 404, 503
Get metadata of an algorithm	GET	/algorithm/{id}	-	Algorithm representation in the requested MIME type	200, 404, 503
Apply an algorithm	POST	/algorithm/{id}	Algorithm-specific parameters	Task URI A task is created that undertakes the asynchronous execution of the algorithm. Upon completion returns the expected result.	200, 400, 404, 503

All algorithm-specific parameters are specified in the representation of the algorithm which can be retrieve upon a GET with the desired MIME type.

Algorithm: JSON Specification		
Field	**Explanation**	**Admissible Values**
`metadata`	An instance of MetaData	`MetaData` objects
`parameter[]`	Set of parameters for the algorithm	Parameter-type objects

Example of the algorithm /algorithm/sb:


```
{
  "object_type":"Algorithm",
  "id":"sb",
  "parameter":[
    {
      "object_type":"Parameter",
      "scope":"OPTIONAL",
      "value":"5",
      "valueType":"Integer",
      "units":"unitless",
      "metadata":{
        "object_type":"MetaData",
        "description":[
          "Number of MC cycles"
        ],
        "identifier":[
          "ad81e445 - 88a4 - 47da - 9e98 - 34d2dc8ab1f5"
        ]
      }
    }
  ],
  "metadata":{
    "object_type":"MetaData",
    "description":[
      "MC algorithm"
    ],
    "identifier":[
      "eb985c6e- f5c2- 4fb4-9186-eaf396e61d0b",
      "sb"
    ]
  }
}
```

4) Task: Tasks undertake asynchronous execution and inform the end user on the progress of their request and its status. Every task therefore is accompanied by a status tag. In the following table we summarise the various possible admissible status values:

Task Status Specification	
Status	**Explanation**
`QUEUED`	The task has been added to a waiting queue and will be eventually submitted to the running queue once a place is freed. The client should check again. The corresponding HTTP status should be 202.
`RUNNING`	The task is being currently executed. Information about its progress are provided. The client should check again in a while following the `Retry-After` header. The accompanying HTTP status header is 202.
`COMPLETED`	The task has completed. A GET request on the task specifying the Accept header text/uri-list points to the generated URI with the result. The accompanying HTTP status header is set to 200.
`CANCELLED`	The task has been cancelled by the end user after applying a DELETE method on the current URI.
`REJECTED`	The task was rejected because both the execution and the waiting pool were full. The client should try again later following the Retry-After directive.

Tasks are created by algorithms upon a POST request and are available in JSON. A JSON representation of a Task holds the following fields:

Task: JSON Specification		
Field	**Explanation**	**Admissible Values**
`metadata`	An instance of MetaData	`MetaData` objects
`taskStatus`	Current Status of the Task	See previous table
`resultURI`	Identifier of the result	URI
`httpStatus`	The accompanying HTTP status	Integer
`duration`	The duration of the execution so far. Takes the value 0 if the task is queued	long
`exp_duration`	The expected duration of the task.	long
`creationTimestamp`	The creation timestamp	long
`error`	An instance of `ErrorReport` with debugging information	Instance of `ErrorReport`

For instance:

```
{
  "object_type":"Task",
  "taskStatus":"COMPLETED",
  "duration":100,
  "exp_duration":5000,
  " httpStatus":200,
  "resultURI":"http:\\\/\\\/server.com:1234\\\/result\\\/1",
  "metadata ":{
    "object_type":"MetaData",
    "title":[
      "My Task"
    ],
    "identifier":[
      "f2990235-9e26-4462-a997-e4c90a0964f8"
```

```
    ]
  },
  "creationTimestamp":1335775580528
}
```

Lists of all tasks are returned in `text/uri-list` format applying a GET on `/task`. Querying is possible using the `status` URL parameter and specifying the desired status (case insensitive).

5) Error: In case an exceptional event occurs, the client is made aware by an Error Report; an object that contains all necessary information for the debugging. The relevant JSON specification is provided in the following table:

ErrorReport: JSON Specification		
Field	**Explanation**	**Admissible Values**
`code`	A standardised error message	String
`message`	Current Status of the Task	String
`toBlame`	Who is to blame: the client, the server or a third-party service.	URI or IP
`debug`	Detailed Debugging Information	String
`httpCode`	The accompanying HTTP code	Integer
`exp_duration`	The expected duration of the task.	long
`trace`	Trace of the current exception	Instance of `ErrorReport`

Here is an example of an ErrorReport serialised in JSON:

```
{
  "object_type":"ErrorReport",
  "code":"Illegal Argument",
  "message":"Execution Failed",
  "debug":"Details about the error...",
  "httpcode":400,
  "toblame":"Client: 10.8.0.1"
}
```

6) Result: Every completed task points to a result. A result is an identifiable set of single-valued or array-valued parameters which are encoded in a loose manner or a set of probability distributions. However, every type of result should implement a given API. How the result looks like, is specified by each algorithm separately however there is a general pattern that is followed. If the result comprises of a set of parameters then these are sequentially encoded according to the following straightforward specification:

Result: JSON Specification		
Field	**Explanation**	**Admissible Values**
`id`	A UUID or other unique identifier for the result	String
`{prm_id}[]`	Any Parameter	Number or Array

Here is an example of how a simple result looks like:

```
{
  "object_type":"Result",
  "LRTP":3.54E-5,
  "Persistency":2.20E-17,
  "PEC":2.11E-5
}
```

For stochastic output, i.e. when the output consists of one or more probability distribution functions, then the result is an array of objects of type ProbabilityDistribution. An example can be found in the appendix. On a result resource, one can apply a GET or a DELETE operation. Here is a short description of the REST API for results:

Result: WS Specification				
Description	**Method**	**URI**	**Result**	**Status**
Get URIs of all results	GET	/result	List of all URIs of available results	200, 404, 500
Get the metadata and actual content of a result in a supported MIME type	GET	/result/{id}	Representation of a result	200, 404, 500
Delete a result from the database of the server	DELETE	/result/{id}	A result is deleted and a simple message is returned to the client explaining the situation. A 500 status code implies that the result cannot be deleted because of some internal error on the server.	200, 404, 500

7) ProbabilityDistribution: A Probability Distribution function is here described by the class of objects ProbabilityDistribution. This consists of the following fields:

ProbabilityDistribution: JSON Specification		
Field	**Explanation**	**Admissible Values**
`average`	A UUID or other unique identifier for the result	Double
`stdev`	The estimated standard deviation of all data points	Double
`points`	All data points used from which the probability distribution function was reconstructed	Array of Double
`intervals`	The PDF is reconstructed as piece-wise constant. The value of this function over each interval is stored under this field.	Array of double
`min`	The minimum observation.	Double
`max`	The maximum observation.	Double
`step`	The step used to partition the interval [min, max] into subinterval of equal breadth over which the PDF admits constant values.	Double
`name`	The name of the variable (more information about the variable can be found in the metadata).	String
`metadata`	Meta-information about the probability distribution function.	MetaData

E. Deterministic & Monte-Carlo Simulations

SBWS can be used to perform both deterministic and Monte-Carlo simulations. For Simplebox-related calculations, the result object contains three fields, namely LRTP (Long-Range Transport), Persistency and PEC (Predicted Environmental Concentration). The expected input parameters for deterministic-mode calculations are the following:

- `emission`: the emitted amount of chemical to the ecosystem in kg.
- `k_OH`: The KOH constant in 1/s.
- `k_photo`: The photo-catalytic degradation constant in 1/s.
- `kdeg.sed`: Bulk degradation rate constant in standard sediment at 25°C in 1/s.
- `kdeg.soil`: Bulk degradation rate constant in standard soil at 25°C in 1/s.
- `kreg.water`: Dissolved phase degradation constant at 25°C in 1/s.
- `MW`: Molecular weight

- `S`: Solubility in water at 25°C
- `Tm`: Melting point in °C.
- `Vp`: Pressure of vapours at 25°C in Pa.
- `Koc`: Partition coefficient between organic carbon and water at 25°C in 1/s.

These parameters are specified in the representation of the algorithm which can be retrieved by a GET request at `/algorithm/sb`. Should the client need to provide the standard deviation for any of the parameters mentioned above, the postfix `_std` should be appended to the name of the respective parameter – for example `Tm_std` stands for the standard deviation of the `S`. This is of course specified in the representation of the algorithm. The parameters for which the standard deviation is specified will be treated as random variables that follow the normal distribution. Other parameters are treated as deterministic. If a parameter's value is set to `MODEL` then a default model will be used to predict the corresponding value and will of course introduce some uncertainty. It is also possible to specify a CADASTER QSAR models using its public identifier. For example, in order to use model 15, provide the value `MODEL15` to the parameter; for instance `Tm=MODEL12`.

There is a currently running version of the web service at 146.107.217.204:8080 that is accessible only from within the HelmholtzZentrum's network.

V. Implementation - GUI

A. User Requirements Analysis

The web interface was designed taking into account the requirements of the CADASTER project. The main idea and the scope behind the design of the developed user interface is to provide a comprehensive and step-by-step online wizard to help risk assessors estimate the risk related to the emission of a chemical to the environment. The user activity workflow presented in figure 6 provides an overview of the steps the user is called to go through within a risk assessment session.

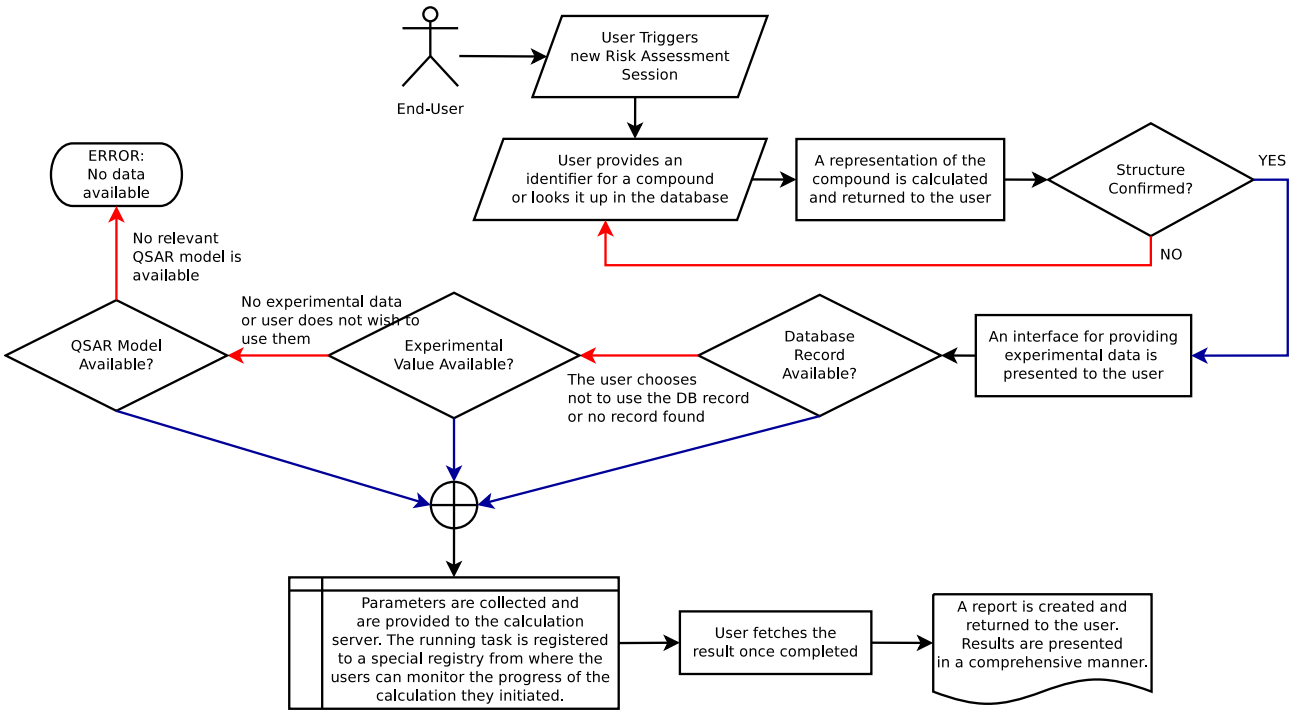

Fig. 6. A workflow of the user activity as specified by the initial requirements of the project The user is presented with a step-by-step wizard.

B. Guided User Interface

The web tool developed for environmental risk assessment and statistical inference is now part of the Cadaster project and can be publicly accessed at qspr-thesaurus.eu. A screenshot of the project's main page is provided in

figure 7.

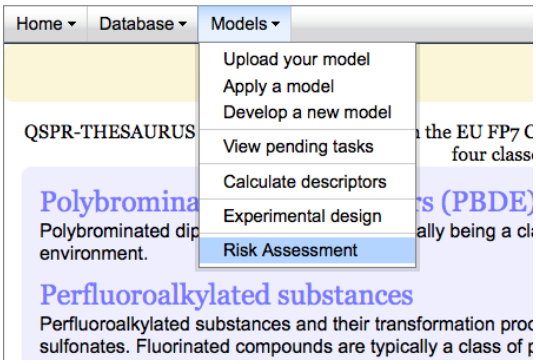

Fig. 7. Probabilistic Environmental Risk Assessment on Cadaster.

Firstly, the user is required to specify the chemical for which the risk assessment calculation will be carried out. This can be done in one of the following ways:

1) By drawing the compound using the molecular editor as in figure 8.
2) By providing the name, CAS registry number or SMILES string of the compound
3) By choosing from a previously prepared *basket* of compounds
4) By uploading a compound using a SD or other relevant file (e.g. MOL or SMI).

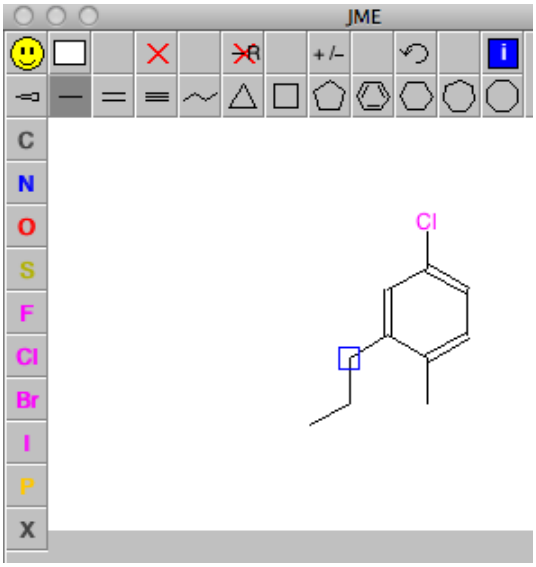

Fig. 8. Screenshot from the web user interface used to draw compounds. JME Molecular editor was developed by Peter Ertl for Novartis.

Users may choose different QSAR models and get prediction values (a nominal value is always accompanied by an reliability estimation) and eventually decide whether the predictions they receive are acceptable on the basis of their uncertainty.

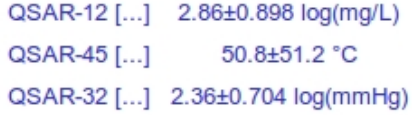

Fig. 9. Screenshot from qspr-thesaurus.eu: Predicted values for Aqueous Solubility, Melting Point and Vapour Pressure. All predictions are accompanied by an estimation of uncertainty.

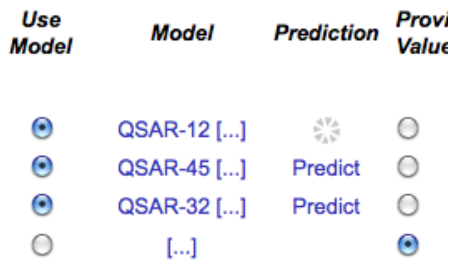

Fig. 10. Screenshot from qspr-thesaurus.eu: User has initiated a new prediction which is in progress.

The resulting probability distribution functions for LRTP, Persistency and PEC (which are of course treated as random variables) are returned to the user. In figure 11 such a PDF is presented for the logarithms of LRTP using 100 random samples.

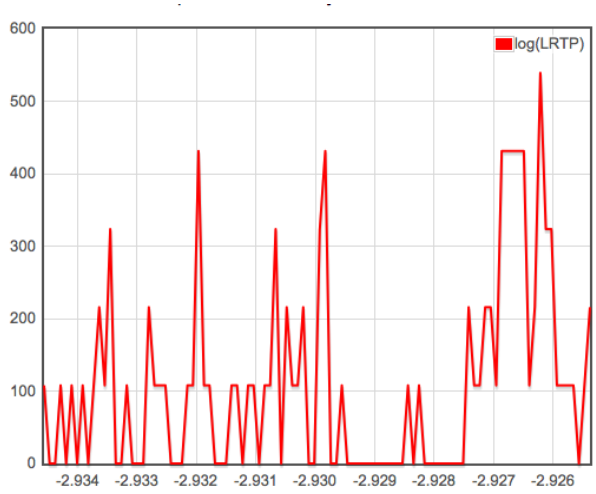

Fig. 11. Probability Distribution Function for log(LRTP) using 100 samples.

VI. Discussion and Conclusions

Risk assessment requires combination of experimental measurements and QSAR/QSPR predictions. Experimental measurements can be either be provided by in-house results that risk assessors have in hand or, most often, by database lookups. All these elements were put together in the web tool that was developed during my fellowship with the Helmholtz research centre. The web tool allows users to search in a large database for available properties, provide their own values and/or use predictive models to estimate certain physicochemical properties for a compound. We strove for the web interface to be as user-friendly as possible: the user is guided step-by-step throughout the whole procedure. Moreover, it is often the case that end-users look down on web applications for the lack of transparency they usually come with. Taken this aspect seriously into account we created a platform where every single piece of information is exposed to the user on request. This way the developed web tool caters for both users that appreciate minimal design and for more advanced users as well.

Following my 3-months fellowship in HMGU, I will continue for another 3 months within the CADASTER project to continue my developments on environmental probabilistic risk assessment. Among my future plans are various improvements of the web interface (improved graphics, interactive material), developments regarding the stability and robustness of the implementation and effect assessment using appropriate software.

References

[1] "Cadaster: Case studies on the development and application of in-silico techniques for environmental hazard and risk assessment," 2009-2012. An FP7-funded EU research project.

[2] K. Solomon, J. Giesy, and P. Jones, "Probabilistic risk assessment of agrochemicals in the environment," in *XIVth International Plant Protection Congress*, vol. 19, pp. 649–655, Sept. 2000.
[3] C.-J. Lee and K. J. Lee, "Application of bayesian network to the probabilistic risk assessment of nuclear waste disposal," *Reliability Engineering & System Safety*, vol. 91, no. 5, pp. 515–532, 2006.
[4] Z. Mohaghegh and A. Mosleh, "Incorporating organizational factors into probabilistic risk assessment of complex socio-technical systems: Principles and theoretical foundations," *Safety Science*, vol. 47, no. 8, pp. 1139–1158, 2009.
[5] S. M. Lynch, *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists*. Statistics for Social and Behavioral Sciences, Springer New York, 2007.
[6] G. R. Terrell, "Getting started," in *Mathematical Statistics* (G. Casella, S. Fienberg, and I. Olkin, eds.), Springer Texts in Statistics, pp. 1–7, Springer New York, 1999. 10.1007/0-387-22769-5_1.
[7] L. Brandes, H. den Hollander, and D. van de Meent, "Simplebox: A nested multimedia fate model for evaluating the environmental fate of chemicals," Tech. Rep. 719101029, National Institute of Public Health and Environmental Protection (RIVM), Bilthhoven, The Netherlands, Dec. 1996.
[8] H. Zhu, A. Tropsha, D. Fourches, A. Varnek, E. Papa, P. Gramatica, T. Oberg, P. Dao, A. Cherkasov, and I. V. Tetko, "Combinatorial qsar modeling of chemical toxicants tested against tetrahymena pyriformis," *J. Chem. Inf. Model.*, vol. 48, 2008.
[9] I. Sushko, S. Novotarskyi, R. Korner, A. K. Pandey, A. Cherkasov, J. Li, P. Gramatica, K. Hansen, T. Schroeter, K.-R. Muller, L. Xi, H. Liu, X. Yao, T. Oberg, F. Hormozdiari, P. Dao, C. Sahinalp, R. Todeschini, P. Polishchuk, A. Artemenko, V. Kuzmin, T. M. Martin, D. M. Young, D. Fourches, E. Muratov, A. Tropsha, I. Baskin, D. Horvath, G. Marcou, C. Muller, A. Varnek, V. V. Prokopenko, and I. V. Tetko, "Applicability domains for classification problems: Benchmarking of distance to models for ames mutagenicity set," *J. Chem. Inf. Model.*, vol. 50, pp. 2094–2111, 2010.
[10] S. Weaver and M. P. Gleeson, "The importance of the domain of applicability in qsar modeling," *Journal of Molecular Graphics and Modelling*, vol. 26, no. 8, pp. 1315–1326, 2008.
[11] J. Jaworska and N. Jeliazkova, "Review of methods to assess a qsar applicability domain," in *QSAR2004*, vol. 1, May 2004.
[12] I. V. Tetko, I. Sushko, A. K. Pandey, H. Zhu, A. Tropsha, E. Papa, T. Oberg, R. Todeschini, D. Fourches, and A. Varnek, "Critical assessment of qsar models of environmental toxicity against tetrahymena pyriformis: Focusing on applicability domain and overfitting by variable selection," *J. Chem. Inf. Model.*, vol. 48, no. 9, pp. 1733–1746, 2008.
[13] H. Sies, "A new parameter for sex education," *Nature*, vol. 332, p. 495, Apr. 1988.
[14] J. Gentle, *Random Number Generation and Monte Carlo Methods*. Statistics and Computing, Springer, 2003.
[15] C. P. Robert and G. Casella, *Méthodes de Monte-Carlo avec R*. Mathematics and Statistics, Springer Paris, 2011.
[16] P. Sopasakis, "Computer-aided drug design and environmental risk assessment," in *3rd Summer of the Marie-Curie ITN programme "Environmental ChemOinformatics"*, (Verona, Italy), June 2012.
[17] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent available partition-tolerant web services," in *ACM SIGACT News*, vol. 33, pp. 51–59, 2002.