INFORMATION SOCIETY TECHNOLOGIES
(IST)
PROGRAMME

OpenMolGRID

SPECIFICATION OF SOFTWARE MODULES FOR DESCRIPTOR CALCULATION AND MODEL DEVELOPMENT AND THEIR GRID INTERFACE COMPONENTS

Contract reference:	**IST-2001-37238**
Document identifier:	**OpenMolGRID-2-D2.1-0101-1-5-DataMining**
Date:	**01/04/2004**
Work package:	**WP2: Molecular Descriptor Generation and QSPR Model Building on the Grid**
Partner:	**UT, UU, NEGRI, FZJ, CGX**
Lead Partner:	**UT**
Document status:	**APPROVED**
Classification:	**PUBLIC**
Deliverable identifier:	**D2.1**

Abstract: Specifications and requirements are given for available software components that will be integrated with the OpenMolGRID project. Application specific metadata necessary for the Grid interfacing is described.

Delivery Slip

	Name	Partner	Date
From	S Sild, A Lomaka	UT	01/04/2004
Verified by	S Sild	UT	01/04/2004
Approved by	GHF Diercksen (TC) R Ferenczi (QE)	OMC CGX	17/04/2004 07/04/2004

Document Log

Issue	Date	Comment	Author
0-0	03/02/2003	First version	S Sild
0-1	26/02/2003	Second stable version; version submitted in second progress and management report	S Sild
1-0	15/09/2003	Third version	A Lomaka, S Sild
1-1	17/09/2003	Fourth version	A Lomaka
1-2	18/09/2003	Fifth version	A Lomaka
1-3	08/10/2003	Sixth version	S Sild
1-4	08/10/2003	Seventh version	S Sild
1-5	30/03/2004	Eighth version	S Sild, A Lomaka

Document Change Log

Issue	Item	Reason for Change
0-1	Minor corrections	Refers to Original Submission to EC
1-0	The structure of the document changed. Theoretical background information moved to D2.4a. Specification updated to include application classes, reviewers comments addressed.	Changes made to address reviewers comments
1-1	Minor corrections	Incorporating review comments by FZJ
1-2	Minor corrections	Incorporating review comments by UU
1-3	Typing errors and references corrected	Internal review process
1-4	Incorporated use cases from a separate file to this main document	Suggestion by M Romberg
1-5	Project information form on page 3, Section 4, and Appendix B	Change of project information; Updated XML schemas and mappings with OpenMolGRID types

Files

Files in this section relate to actual storage locations on the BSCW server located at https://hermes.chem.ut.ee/bscw/bscw.cgi. The URL below describes the location on BSCW from the root OpenMolGRID directory

Software Products	User files / URL
Word 2000/XP	OpenMolGRID/Workpackage 2/Deliverables/ OpenMolGRID-2-D2.1-0101-1-5-DataMining

Project information

Project acronym:	OpenMolGRID
Project full title:	Open Computing GRID for Molecular Science and Engineering
Proposal/Contract no.:	IST-2001-37238
European Commission:	
Project Officer:	Annalisa BOGLIOLO
Address:	European Commission - DG Information Society F2 - Grids for Complex Problem Solving B-1049 Brussels Belgium
Office	BU31 4/79
Phone:	+32 2 295 8131
Fax:	+32 2 299 1749
E-mail	annalisa.bogliolo@cec.eu.int
Project Coordinator:	Mathilde ROMBERG
Address:	Forschungszentrum Jülich GmbH ZAM D-52425 Jülich Germany
Phone:	+49 2461 61 3703
Fax:	+49 2461 61 6656
E-mail	m.romberg@fz-juelich.de

Contents

1. Introduction

1.1. Purpose and scope

The aim of this document is to formulate user requirements for the data mining environment, and to specify data formats for individual software modules and protocols for their interaction.

1.2. Document Overview

The data mining environment provides seamless access to various descriptor calculation and model development applications, as well as automated workflow support that is essential for multistep modelling problems like QSPR/QSAR. The current document describes how the UNICORE infrastructure helps to achieve these goals and to hide the internal details of the Grid from users with (bio)chemistry background. The brief overview about the relevant theoretical background information is given in Deliverable D2.4a [1].

1.3. Document Structure

In addition to this section the document contains the following sections:

- Section 2 describes the main objectives of the Grid integration
- Section 3 covers the user and functional requirements of data mining
- Section 4 gives an overview of UNICORE applications. Subsections contain a brief description, application neutral input/output formats and requirements of each application
- Section 5 describes data exchange by UNICORE communication protocol
- Section 6 contains references to relevant documents
- Section 7 contains list of abbreviations used in this document
- Appendix A contains describes use cases
- Appendix B contains XML schema definitions for data formats

2. General Description

The Grid infrastructure for molecular design problems must be able to support the coordinated use of several distributed software modules. In order to integrate the software modules with the Grid, the following technological aspects need to be considered:

- Standard interfaces and data formats for tasks of software packages. This is necessary in order to reuse the same interface for several software packages (extensibility) and to allow to easily make changes in software packages (flexibility). Methods to handle extensibility and flexibility within the Grid infrastructure are described in section 4 for each molecular modelling task.
- Data exchange between integrated applications to provide secure access to data and software resources in a uniform manner. The relevant architecture is described in OpenMolGRID Deliverable 4.5a [2].

3. Requirements for the Data Mining Environment

The data mining (DM) environment within the OpenMolGRID system will comprise a set of domain specific tools that are built on top of the UNICORE infrastructure. The DM environment within the OpenMolGRID project is intended for academic, scientific and industrial end-users from chemical and pharmaceutical industry. The end-users usually have some background knowledge in various domains of chemistry, biochemistry, computational chemistry, and statistical analysis. Usually, a user is highly skilled specialist in one or two of above-mentioned domains, and has some basic knowledge about others. In addition, end-users are expected to have good skills with computers. The users will expect to employ a set of tools that permits them securely and seamlessly access and use globally distributed resources and systems relevant to QSPR/QSAR modelling.

The main use cases are described briefly in 3.1 and more formally in Appendix A.

3.1. Use Cases

In the highest level, the use cases relevant to the DM environment can be classified according to two major goals, as follows:

- Development of QSPR/QSAR models
- Prediction of chemical properties or biological activities

Both these high level goals include a number of common sub-cases, e.g. the optimisation of molecular structures, quantum-chemical calculations, calculation of molecular descriptors, etc. In addition, some individual sub-cases may have separate goals on their own. For instance, the end-user may use the system only as an interface to quantum chemical calculations or statistical analysis. Some use cases related to the data management, e.g. importing data from a data warehouse, are closely connected to WP1 (Grid Data Warehousing).

4. UNICORE Applications

Integration of molecular design software packages with the Grid will be accomplished by using the UNICORE infrastructure. Software packages will be wrapped by UNICORE applications in order to achieve independence from changes in underlying software packages (flexibility) and unified access to different software applications under the same task (extensibility). The application is controlled by a UNICORE client component or plugin that includes containers and GUI panels. The application is described to a UNICORE client by the metadata file containing information about different tasks provided by the application, control parameters and other information. The wrapper is also responsible for converting the application specific formats to some application neutral format. This helps to reuse the same plugin for different software packages.

Methods are also being developed for grouping the tasks into more complex workflow. This is especially important for QSPR/QSAR modelling which normally consists of several separate tasks. The workflow is controlled by another plugin called "meta plugin".

A more detailed overview of Grid integration has been given in Deliverable 4.5a [2] and OpenMolGRID Plugin development guide [3].

The following sections provide a general description of application specific interfaces, descriptions of application neutral input/output formats, and metadata for software packages that will be integrated with OpenMolGRID. The detailed description about the application specific programs is given in Deliverable D2.4b [4].

4.1. Structure conversion from 2D to 3D

The structure conversion application will convert 2D representation of molecular structure to 3D representation. Within OpenMolGRID project the MOLGEO software will be integrated with the DM environment.

4.1.1. Input Formats

The structure conversion task takes a list of molecular structures as input. The XML schema for the structure list is given in Section 9.1 (StructureList) together with mappings to OpenMolGRID data types [5], which are used in data sources that are provided by WP1. The molecular structures in the structure list are stored in the Chemical Markup Language (CML) format. In addition, the MDL's MOL file format is supported, as it is very widely supported format by chemical software.

4.1.2. Output Formats

The output format from the structure conversion task is same as the input format (see Section 9.1 for the XML schema definition).

4.1.3. Application Specific Metadata

Parameter Name	Type	Description
Time limit	Integer	Time limit for calculation in seconds. Default is zero, which indicates that execution time is not limited.
Tolerance	String	The convergence limit for structure optimisation. Possible values are: "lowest", "low", "medium", "high", "highest". Default is "medium".
Verbose	String	Verbose information about the execution of the program which is useful for diagnostics. Possible values are: "true" and "false". Default is "false".

4.1.4. Requirements

2D to 3D conversion is necessary whenever user decides to calculate descriptors that require 3D molecular information. Such descriptors include, e.g., various surface area related descriptors and quantum chemical descriptors. The need for conformational search depends on the size and flexibility of molecules and types of descriptors that are used for modelling.

4.2. Descriptor calculation

The molecular descriptor calculation application will provide generic wrapper for descriptor calculation software programs. For the OpenMolGRID project the MDC module from the CODESSA PRO program is adapted.

4.2.1. Input Formats

The descriptor calculation task accepts XML formats that are defined in Sections 9.1 (StructureList) and 9.2 (SemiempiricalOutput). The molecular structures must have 3D coordinates defined. Normally, the output from 2D to 3D conversion or semi-empirical quantum chemical calculations is used as an input for the descriptor calculation task in the OpenMolGRID workflows.

4.2.2. Output Formats

The XML schema for the descriptor calculation output is defined in Section 9.3 (DescriptorFile). This file contains metadata about calculated descriptors, descriptor identification numbers, and calculated descriptor values.

4.2.3. Application Specific Metadata

Parameter Name	Type	Description
QC/TD suppress	Integer	Flag to indicate whether the quantum chemical and/or thermodynamical descriptor calculation should be avoided. This is only relevant when output from semi-empirical output is used. Possible values: 1 – calculate all descriptors 2 – don't calculate thermodynamical descriptors 3 – don't calculate quantum chemical and thermodynamical descriptors

4.2.4. Requirements

The calculation of molecular descriptors for chemical structures should be possible both in interactive and batch mode. The calculation in interactive mode will provide graphical tools that allow user to choose list of structures from available data sources, define list of molecular descriptors to be calculated, and choose target site for the calculation. The current list of descriptors is defined by selecting descriptors from the list of all available descriptors, where following information will be visible for the user:

- Descriptor name
- Information about the descriptor (e.g. equation, reference to literature)
- Calculation software name and version number
- Type information (e.g. topological, electrostatic, quantum-chemical, etc.)
- Sub-type (type of partial charges used, AM1/PM3, etc.)

The molecular descriptor selection tool will provide several sorting options for the display of available descriptors, such as by the descriptor name, by the calculation software, by the type information, by

the sub-type information, and their combinations. Molecular descriptors can be marked as selected by the descriptor name, by the type information, by the sub-type, by the calculation software, or by the sub-string.

4.3. QSPR/QSAR model building

The QSPR/QSAR model development module (MDA) will be integrated from the CODESSA PRO program. The following types of QSPR/QSAR models will be available for the user:

- Multilinear Regression Models (MLR)
- Partial Least Squares (PLS)
- Principal Components Regression (PCR)
- Artificial Neural Networks (ANN)

4.3.1. Input Formats

The model building task needs input files with descriptor and property values. The XML schema for representing descriptor values is defined in Section 9.3 (Descriptor File). The XML schema for the property values is defined in Section 9.4 (Property File).

4.3.2. Output Formats

The XML schema for the QSPR/QSAR models is defined in Section 9.5 (ModelBuildingOutput). Originally the Predictive Model Markup Language (PMML) format was planned for representing QSPR/QSAR models. However, the PMML format is not suitable for representing PCR and PLS models.

4.3.3. Application Specific Metadata

4.3.3.1. Heuristic method PRO.

Parameter Name	Type	Description
F_order	Float	Order of Fisher criteria. Default value 1.
R2_order	Float	Order of squared correlation coefficient. Default value1.
N_order	Float	Order of number of descriptors. Default value 1.
S2_order	Float	Order of squared standard error. Default value –1.
ND_order	Float	Order of number of datapoints. Default value –1.
Codessa_compatible	Boolean	“True” if compatible with old Codessa, “false” otherwise.
Extended	Boolean	“True” if using extended testing of 1-parameter linear model, “false” otherwise. Default value “false”.
Min_R2	Float	Minimum squared correlation coefficient. Default value 0.1.
Max_stderr	Float	Maximum standard error. Default value 0.
Min_student	Float	Minimum Student criterion of descriptor. Default value 1.5
Min_R2_LOO	Float	Minimum squared correlation coefficient (leave one out crossvalidation). Default value 0
Min_R2_LMO	Float	Minimum squared correlation coefficient (leave many out crossvalidation). Default value 0.

Min_W1	Float	Minimum weight of the 1-parameter correlation. Default value 0.
Sort	Int	Sorting criterion. Possible values: 1 - R2, 2 - F, 3 - s2, 4 - R2cvoo, 5 – w Default value 1.
Pairwise	Boolean	"Yes" if calculating pairwise descriptor intercorrelations, "false" otherwise. Default value "true".
Max_IC	Float	Maximum value of descriptors' IC for including into IC matrix. Default value 0.99.
Max_IC_startup	Float	Maximun value of descriptors' IC for generating 2-parameters correlation in startup pool. Default value 0.5.
Startup_2par	Integer	Criteria for including 2-parameters correlation in startup pool. Has one of the following values: 0 - include all, 1 - Student's criteria (for both descriptors), 2 - weight increasing. Default value 1.
Memsize	Integer	Size of in-memory model pool. Default value 100000.
Max_expand	Integer	Maximum of descriptors for developed models (for expanding). Default value 5.
Pool_reuse	Boolean	Reuse of correlation pool (deleting of bad correlations). "True" if permitted, "false" otherwise. Default value "false"
Max_it	Integer	Maximum number of iterations. Default value 1000.
T_limit	Integer	Time limit for first stage in seconds. Default value 3600.
Frac_LOO	Float	Fraction of training set for leave-many-out crossvalidation. Default value 0.8.
N_cross	Integer	Number of crossvalidation tests. Default value 1000
N_rand	Integer	Number of randomization tests. Default value 1000.
Max_R2_IC	Float	Maximum descriptor intercorrelation. Default value 0.5.
Max_R2_OO	Float	Maximum percent of lost in R2cvOO in comparison to R2. Default value 5.
Max_R2_MO	Float	Maximum percent of lost in R2cvMO in comparison to R2. Default value 10.
Max_rand_unsuccessful	Float	Maximum fraction of unsuccessful randomization tests. Default value 0.001.
N_corr	Int	Number of correlations per number of descriptors. Default value 10.
Save	Boolean	Save the best correlations for every number of descriptors? Default value "true".

4.3.3.2. Principal component regression.

Parameter Name	Type	Description
Conv	Float	PCA convergence criterion. Default value 0.01.
Method_PCA	Integer	Cutoff criterion for PCA method. Can have one of the following values: 1 – absolute value of the residual matrix 2 – real error

		3 – imbedded error 4 – factor indicator function 5 – cumulative percentage of the variation 6 – Kaiser's rule 7 – Broken stick model 8 – Catell rule Default value 2.
Desc_scales	Integer	Input data pretreatment. Can have any of the following values. 0 - the natural initial matrix is used as the input to PCA 1 - the natural scales centered and then normalized are used as the input to PCA 2 - the natural scales normalized and then centered are used as the input to PCA Default value 2.
Eps	Float	Best regression convergence limit. Default value is 0.1.
R2min	Float	R2(min) acceptable for 3-parameter correlation. Default value 0.6.
Delta	Float	Delta[R2(min)] for successive correlations. Default value 0.02.

4.3.3.3. Partial least squares analysis.

Parameter Name	Type	Description
Max_res	Float	Maximum residual sum of squares. Default value 0.01.
Max_compo	Integer	Maximum number of components to extract. Default value 5.

4.3.4. Requirements

The QSPR/QSAR model building needs a dialog to select particular method for the model development. This dialog should include the list of available methods. In order to simplify the selection of suitable algorithm, the following information should be available for each algorithm:

- Method name
- Description of a method (e.g. link to help file or reference to journal article)
- Author
- Copyright
- Calculation software name and version number
- List of sites, where calculation module is installed

After the method for the QSPR/QSAR model development is selected, the control parameters should be defined. For each method, dialog for the parameter definition should be available. This dialog should provide the following information about each parameter that can be adjusted by the user:

- Parameter name
- Description of a parameter
- Range of allowed values

In addition, each parameter should have a reasonable default value and there should be an option to restore default values of parameters.

It is also necessary to receive feedback from the model building process. Most programs write progress information to a log file and often write intermediate results to temporary files. It should be possible to display these log files in a window and the visualisation programs should have access to the temporary files. The ability to start, stop, suspend or restart model building process will be also important feature of the system. The job control provided by the UNICORE infrastructure can be utilised to achieve this.

The output from the QSPR/QSAR treatment must include information about the data set that was used to develop a model. The enables the selection and visualisation of structures from a data set and to view their descriptor values and to view their experimental and calculated property/activity values. The developed QSPR/QSAR models will be characterised by statistical parameters such as: correlation coefficients, standard deviation, root mean square error, etc. The model visualisation tool should provide 2D plot of calculated vs. experimental property/activity values. This plot should provide facility to select individual data points and display its structure name, and highlight outliers from the plot. The model visualisation tool will also show the mathematical representation of a QSPR/QSAR model (e.g. regression coefficients in the case of MLR models or neural network weights in the case of ANN model).

4.4. Semi-empirical quantum chemical calculations

The semi-empirical quantum chemical calculations will be performed using MOPAC software. The integration of MOPAC is straightforward, since the communication with this program is file based. Codessa PRO uses standard sets of keywords in input files are used to control calculation. Parameters that can be varied are described in metadata section.

4.4.1. Input Formats

The XML schema for the input to semi-empirical calculation task is defined in Section 9.1 (StructureList). The molecular structures must have 3D coordinates defined. Usually, the output from the 2D to 3D conversion task is used as input for this task in OpenMolGRID workflows.

4.4.2. Output Formats

The XML schema for the output from semi-empirical calculation task is defined in Section 9.2 (SemiempiricalOutput).

4.4.3. Application Specific Metadata

Parameter Name	Type	Description
TLimit	Integer	Time limit for calculation in seconds. Default is zero, which indicates that execution time is not limited.
Calc_type	Integer	Calculation type. Can be one of the following: 1 – Optimization 2 – Thermodynamic calculation
Parameterization	String	Can be one of the following: AM1 – Austin model 1 PM3 – parametric method 3

4.4.4. Requirements

Semi-empirical calculations are prerequisites for the calculation of the quantum chemical descriptors. They are also needed for the calculation of thermodynamical descriptors.

5. Communication Protocols

The OpenMolGRID modules or components will be integrated in the UNICORE infrastructure. Data exchange between UNICORE components is file based, thus UNICORE needs access to the file system at the local and at the target host. The integrated applications will be used through UNICORE client plugins for the specific "Software" or "Application" resources, which are offered by the UNICORE Virtual Site (Vsite). The application has to be installed on the actual target system and the UNICORE server needs to have it defined in the IDB (Incarnation Data Base). The IDB contains the Vsite specific resource information and the translation table for mapping the abstract UNICORE job into the concrete job for the specific Virtual Site (target system).

The calculation module calls are prepared in the application specific plugin for the module in the UNICORE user interface. The user selects a target site for the calculation. When the application is not available at the selected Vsite, the user gets a respective warning. The request for the calculation is handed over via the Network Job Supervisor (NJS, which does the translation into the concrete job according to its IDB) to target system interface (TSI). The TSI hands over the execution statement to the operating system or to the resource management system.

The user interaction during the data mining process (e.g. checking or visualising the intermediate results from a calculation, cancelling the calculation, etc.) is also expected. The visualisation of intermediate results will rely on temporary files that are generated during the calculations and then accessed by the visualisation software. The UNICORE job control facilities can be used to cancel running calculations.

6. References

1. A Lomaka, U Maran, Description of the Quantitative Structure Property/Activity Relation Model: Model Building and Application, OpenMolGRID-2-D2.4a-0102-1-0, **2003**.
2. M Romberg, B Schuller, Description of the OpenMolGRID Grid Architecture, Security Architecture, and Infrastructure and the Deployment of the Testbed, OpenMolGRID-4-D4.5a-0110-1-1, **2003**.
3. B Schuller, M Romberg, OpenMolGRID plugins: Guidelines for development,OpenMolGRID-4-TED-0107-0-5, **2003**.
4. A Lomaka, Description of Codessa Pro Modules (and Mopac), OpenMolGRID-2-D2.4b-0103-1-0, **2003**.
5. B Schuller, D McCourt, P Hliva, OpenMolGRID Data Types, OpenMolGRID-1-TED-0126-0-2-OMGDataTypes, **2004**.

7. Terminology / Glossary

ANN	Artificial Neural Network
CDR	Custom Data Repository is permanent data source or data storage, which is located on local or remote site.
DM	Data mining
DW	Data warehouse
MLR	Multi-Linear Regression
PCR	Principal Component Regression
PLS	Partial Least Squares
QC	Quantum-Chemical
QSAR	Quantitative Structure-Activity Relationship
QSPR	Quantitative Structure-Property Relationship
UNICORE	Uniform Interface to Computer Resources

8. Appendix A: Use Cases

8.1. Build QSPR/QSAR model (data mining)

8.1.1. Use case: **Build a multilinear regression (MLR) model**

Scenario: Develops QSPR/QSAR models by running the multilinear regression (MLR) method. User selects one of the available algorithms with default parameter settings and saves some of the produced models

Preconditions:
1. Property or activity selected from the CDR
2. Molecules list selected from the CDR
3. Descriptors list selected from the CDR
4. Default values for model building software

Trigger: User selects MLR from QSPR/QSAR menu

Description:
1. MLR dialog shows list of available algorithms for model development
2. User selects Heuristic algorithm
3. MLR dialog shows parameters for Heuristic algorithm with their default values
4. User accepts default values
5. MLR dialog checks that all selected structures have property values available (Use case: Handle missing property or activity)
6. MLR dialog checks that all selected descriptors are available for every structure in the molecules list (Use case: Handle missing descriptor values)
7. MLR dialog shows list of target systems for QSPR/QSAR model development
8. User selects target system for calculations
9. Report window gives a user an overview of the progress of the calculation
10. At the end of the calculation, user will be presented a list of MLR models together with their statistical criteria
11. User selects some of the models
12. User saves selected MLR models

Extensions:

Variations:

2a User selects Best Multilinear Regression (BMLR) algorithm

4a User changes default values

8a User selects local machine for calculation

8b User selects remote site for calculation

Postconditions: New MLR models are saved in the CDR

8.1.2. Use case: **Build a partial least squares (PLS) model**

Scenario: Develops QSPR/QSAR models by running the partial least squares method with default parameter settings and saves some of the produced PLS models.

Preconditions:
1. One or more properties or activities selected from the CDR
2. Molecules list selected from the CDR
3. Descriptors list selected from the CDR
4. Default parameters for PLS software

Trigger: User selects PLS from QSPR/QSAR menu

Description:
1. PLS dialog shows list of parameters with their default values
2. User accepts default values
3. PLS dialog checks that all selected structures have property values available (Use case: Handle missing property or activity)
4. PLS dialog checks that all selected descriptors are available for every structure in the molecules list (Use case: Handle missing descriptor values)
5. PLS dialog shows list of target systems for QSPR/QSAR model development
6. User selects target system for calculations
7. Report window gives a user an overview of the progress of the calculation
8. At the end of the calculation, user will be presented a list of PLS models together with their statistical criteria
9. User selects some of the models
10. User saves selected PLS models

Extensions:

Variations:

2a User changes default values

6a User selects local machine for calculation

6b User selects remote site for calculation

Postconditions: New PLS models are saved in the CDR

8.1.3. Use case: **Build a principal component regression (PCR) model**

Scenario: Develops QSPR/QSAR models by using the principal component regression method with default parameter settings and saves some of the produced PCR models.

Preconditions:
1. Property or activity selected from the CDR
2. Molecules list selected from the CDR
3. Descriptors list selected from the CDR
4. Default parameters for the PCR model development software

Trigger: User selects PCR from QSPR/QSAR menu

Description:
1. PCR dialog shows list of parameters with their default values
2. User accepts default values
3. PCR dialog checks that all selected structures have property values available (Use case: Handle missing property or activity)
4. PCR dialog checks that all selected descriptors are available for every structure in the molecules list (Use case: Handle missing descriptor values)
5. PCR dialog shows list of target systems for QSPR/QSAR model development
6. User selects target system for calculations
7. Report window gives a user an overview of the progress of the calculation
8. At the end of the calculation, user will be presented a list of PCR models together with their statistical criteria
9. User selects some of the models
10. User saves selected PCR models

Extensions:

Variations:

2a User changes default values

6a User selects local machine for calculation

6b User selects remote site for calculation

Postconditions: New PCR models are saved in the CDR

8.1.4. Use case: **Build a artificial neural network (ANN) model**

Scenario: Develops non-linear QSPR/QSAR models by using the artificial neural networks and saves some of the produced ANN models.

Preconditions:
1. One or more properties or activities selected from the CDR
2. Molecules list selected from the CDR
3. Descriptors list selected from the CDR
4. Default parameters for the ANN model building software

Trigger: User selects ANN from QSPR/QSAR menu

Description:
1. ANN dialog checks that all selected structures have property values available (Use case: Handle missing property or activity)
2. ANN dialog checks that all selected descriptors are available for every structure in the molecules list (Use case: Handle missing descriptor values)
3. ANN dialog shows list of parameters with their default values
4. User accepts default values
5. ANN dialog displays tool for dividing structures to training and validation sets
6. User creates training and validation sets
7. ANN dialog shows list of target systems for QSPR/QSAR model development
8. User selects target system for calculations
9. Report window gives a user an overview of the progress of the calculation
10. At the end of the calculation, user will be presented a list of ANN models together with their statistical criteria
11. User selects some of the models
12. User saves selected ANN models

Extensions:

Variations:

4a User changes default values

8a User selects local machine for calculation

8b User selects remote site for calculation

Postconditions: New ANN models are saved in the storage

8.2. Predict property or activity values from molecular structure

8.2.1. Use case: **Predict property or activity values by QSPR/QSAR model**

Scenario: User selects a QSPR/QSAR model and uses this model to predict properties for a selected list of structures.

Preconditions:
1. One or more QSPR/QSAR models stored in the CDR.
2. One or more molecules lists stored in the CDR

Trigger: User selects "QSPR/QSAR Prediction" from the Prediction menu

Description:

1. Prediction tool displays QSPR/QSAR model selection tool with a list of available models with short description about each model (property/property name, statistical parameters, model type, number of compounds and descriptors,).
2. User selects one QSPR/QSAR model for prediction
3. Prediction tool displays molecules lists that are available in the CDR
4. User selects a molecules list (prediction set) for the prediction
5. Prediction tool extracts list of descriptors from the QSPR/QSAR model and checks that these descriptors are available for all molecules in the prediction set.
6. All the necessary descriptors are available and prediction of property/activity values is started
7. Prediction tool extracts prediction module information from the QSPR/QSAR model and input data in suitable format for prediction module is prepared.
8. Prediction tool runs prediction module
9. Prediction tool displays spreadsheet with structure names, predicted property/activity values and descriptor values
10. User saves prediction results to a tab delimited file

Extensions:

4a User creates a new molecules list (Use case: Create a new molecules list)

6a Some or all structures have no descriptors available, offer option to calculate missing descriptors

6a1 User decides to calculate missing descriptors

6a2 Structures with missing descriptors are selected and descriptors are calculated (Use case: Calculate molecular descriptors)

Variations:

Postconditions: Molecules in the CDR are updated with calculated property values

8.2.2. Use case: **External validation of the QSPR/QSAR model**

Scenario: User selects a QSPR/QSAR model and uses external testing set for the validation of the selected model.

Preconditions:
1. One or more QSPR/QSAR models stored in the CDR
2. One or more molecules lists with experimental propery/activity values defined in the CDR

Trigger: User selects "QSPR/QSAR model validation" from the Prediction menu

Description:
1. Prediction tool displays QSPR/QSAR model selection tool with a list of available models with short description about each model (property/property name, statistical parameters, model type, number of compounds and descriptors,).
2. User selects one QSPR/QSAR model for validation
3. Validation tool displays molecules lists that are available in the CDR
4. User selects one list (external testing set) for the prediction
5. Validation tool extracts list of descriptors from the QSPR/QSAR model and checks that these descriptors are available for all the structures in the prediction set.
6. All the necessary descriptors are available and prediction of property/activity values is started
7. Validation tool extracts prediction module information from the QSPR/QSAR model and input data in suitable format for prediction module is prepared.
8. Validation tool runs prediction module
9. Validation tool displays statistical parameters about the prediction set. In addition, spreadsheet with structure names, experimental and predicted property/activity values, and prediction errors is displayed.
10. Statistical parameters about the prediction set are saved with selected QSPR/QSAR model

Extensions:

4a User creates a new molecules list (Use case: Create a new molecules list)

6a Some or all structures have no descriptors available, offer option to calculate missing descriptors

6a1 User decides to calculate missing descriptors

6a2 Structures with missing descriptors are selected and descriptors are calculated (Use case: Calculate molecular descriptors)

Variations:

Postconditions:
1. Molecules in the CDR are updated with calculated property values
2. QSPR/QSAR model in the storage updated with information about the prediction set

8.3. Calculate Molecular Descriptors

8.3.1. **Use case**: Select and calculate molecular descriptors

Scenario: User has created a molecules list and decided to calculate some molecular descriptors for given structures. User will make a selection from a list of available descriptors, which will be then calculated for every structure in molecules list and saved to the storage.

Preconditions:
1. Molecules list selected from the CDR
2. Descriptors lists available in the CDR

Trigger: User selects "Calculate descriptors"

Description:
1. Descriptor calculation (DC) manager is started and provides user with selection of available lists of descriptors.
2. User selects one list of descriptors
3. DC manager checks that molecular structures are in required representation
4. All structures have required data available for descriptor calculation
5. Molecular descriptor are calculated (**Use case:** Calculate molecular descriptors)

Extensions:

2a User creates a new list of descriptors (**Use case:** Create a new list of descriptors)

4a Some structures have missing 3D geometries and automatic conversion from 2D to 3D is performed (Use case: **Convert 2D structure to 3D structure**)

4b Some structures do not have MOPAC output data available and MOPAC calculation is initiated (Use case: **Quantum chemical calculation with MOPAC**)

Variations:

Postconditions: Calculated descriptor values stored in the CDR

8.3.2. Use case: **Calculate molecular descriptors**

Scenario: User has decided to build a new QSPR/QSAR model or to use existing model for the prediction. Both these tasks require molecular descriptors to be available for every selected molecule. However, some or all selected molecules have no descriptor values calculated. After this use case, every selected molecule will have required molecular descriptors calculated.

Preconditions:
1. Molecules list selected from the CDR
2. Descriptors list defined

Trigger: Missing molecular descriptors for some or all molecules

Description:
1. Descriptor calculation (DC) manager is started
2. Find out which DC module can calculate given descriptors in the descriptors list. Result for each descriptor will contain information relevant DC module (module name, version, executable name and list of acceptable file formats)
3. DC manager prepares input files with molecules list and descriptors list for relevant DC module(s)
4. Display list of target systems for descriptor calculation
5. User selects target system
6. DC manager transfers input files to target system and initiates descriptor calculation. When calculation is finished, results are transferred back and input files are removed.
7. Attach calculated descriptor values to molecules and save them to storage

Extensions:

Variations:

3a 2D representation is required and 2D structures are available

3b 2D representation is required, but 3D structures are available

3c 3D representation is required, but 2D structures are available (Use case: **Convert 2D structure to 3D structure**)

3d Output form MOPAC is required and output file is available in the storage

3e Output form MOPAC is required, but 3D structure is available (Use case: **Quantum chemical calculation with MOPAC**)

3f Output form MOPAC is required, but 2D structure is available

3f1 Perform 2D to 3D conversion (Use case: **Convert 2D structure to 3D structure**)

3f2 Perform MOPAC calculation (Use case: **Quantum chemical calculation with MOPAC**)

Postconditions: Calculated descriptor values stored in the CDR

8.4. Data management in data mining environment

8.4.1. Use case: Manually add structures to the CDR

Scenario: User uses graphical interface to sketch structures of molecules and optionally enter associated property or activity values.

Preconditions: CDR available

Trigger: User selects "Add structures" in data management tool

Description:
1. Structure entry tool provides facilities, where user can sketch molecular structure and enter its name and associated property values
2. User sketches molecular structure.
3. User enters structure name
4. User optionally enters property names and values
5. User saves structure to the CDR

Extensions:

Variations:

Postconditions: Molecule saved to the CDR

8.4.2. Use case: **Import structures from .SD file**

Scenario: User selects a file in .SD file format. Molecular structures from this file are imported and stored to the CDR. If .SD file contains property or activity data, then this information is attached to imported structures.

Preconditions:
1. CDR available
2. .SD file available

Trigger: User selects "Import .SD file" from data management tool

Description:
1. The .SD file import tool displays file selection dialog.
2. User selects one .SD file
3. .SD file import tool asks user to specify the type (2D or 3D) of structure representation
4. User specifies type of structure representation
5. The .SD file import tool opens the selected .SD file, imports all the structures together with property/activity values if they are present.
6. Result is stored to the CDR

Extensions:

4a If structure type is 3D, then source of 3D structure is asked

5a If molecule has enumeration information in .SD file, then this information is not stored to CDR

6a Handling of duplicated structures

Variations:

Postconditions: Molecules (with or without associated property/activity values) stored in the CDR

8.4.3. Use case: **Create a new molecules list**

Scenario: User defines a new molecules list, which is a container for holding structures. Structures can have several representations (e.g. chemical name, CAS number, 2D structure, 3D structure) and various characteristics attached to it (e.g. property or activity values, descriptor values, results from quantum-chemical calculations, etc.).

Preconditions: CDR with one or more structures available

Trigger: User selects "Define molecules list"

Description:
1. Molecules list creation tool provides facility to specify name for a new molecules list and a comment string about the molecules list.
2. User enters name and comment string
3. Empty molecules list is created (additional information about creation time and user who created it is stored to the molecules list)
4. Structure selection tool appears, providing different options to add new molecules into the molecules list
5. User selects molecules for the molecules list
6. User saves molecules list
7. Molecules list is saved to the CDR for future processing

Extensions:

Variations:

Postconditions: Molecules list stored in the CDR

8.4.4. Use case: **Create a new list of descriptors**

Scenario: OpenMolGRID system has a list of all descriptors that have been defined. In addition, sub-lists exist, where descriptors have been classified according to their type, calculation module, etc. User defines a new descriptors list by selecting one or more relevant descriptors, by selecting one sub-list or by combining several sub-lists.

Preconditions: CDR with one or more descriptors available

Trigger: User selects "Define descriptors list"

Description:

1. Descriptors list creation tool provides facility to specify name and comment string for a new descriptors list.
2. User enters name and comment string for the descriptors list
3. Empty descriptors list is created (additional information about creation time and user who created it is stored to the list)
4. Descriptor selection tool will be displayed.
5. User selects one or more descriptors
6. User saves descriptors list
7. Descriptors list is saved to the storage for future processing

Extensions:

Variations:

5a User selects descriptors by name
5b User selects descriptors by type
5c User selects descriptors by descriptor calculation module name (e.g. select all CODESSA descriptors)

Postconditions: Descriptors list in the storage

8.4.5. Use case: **Select property or activity**

Scenario: User selects one property or activity from the CDR that will be used to develop a QSPR/QSAR model.

Preconditions: Property or activity data in the CDR

Trigger: "Select property" option in model development or prediction tools

Description:
1. Property selection tool will provide list of available property and activity names and option to select one data source.
2. User selects one property or activity from the CDR
3. Property or activity is selected

Extensions:

Variations:

Postconditions: Property or activity selected

8.4.6. Use case: Handle missing property or activity values

Scenario: Molecules that are used for the development of a QSPR/QSAR model need to have all property values defined. This use case checks that the molecules list has all property values defined and offers some methods to handle structures with missing property values.

Preconditions: Molecules list is available in the CDR

Trigger:

Description:
1. Structures with missing property/activity values are detected. User is provided with options to handle missing values.
2. User selects option query a DW for missing property/activity data.
3. System will prepare query for structure search and initiates structure search in a DW (Use case: Retrieve property/activity values from a DW) Property/activity data that is found will be attached to the molecules list.
4. User will type missing property values for few structures.
5. System will add this data to molecules list
6. User will select option to remove remaining structures from the molecules list that have missing property data
7. Structures with missing property/activity data are deleted from a molecules list

Extensions:

Variations:

Postconditions: Molecules list where all structures have all the property/activity values defined

8.4.7. Use case: **Handle missing descriptor values**

Scenario: Molecules that are used for the development of a QSPR/QSAR model must have molecular descriptor values defined. This use case checks that the molecules list has all descriptor scales properly defined and offers several methods to handle structures with missing descriptor values.

Preconditions: Molecules list and descriptors list is available in the CDR

Trigger: Some molecules have required descriptor values missing

Description:

1. Structures with missing property/activity values are detected. User is provided with options to handle missing values.
2. User selects option to calculate missing molecular descriptors (Use case: Calculate molecular descriptors). Step 1 is repeated to see if there are some missing descriptor values left.
3. Some structures have still some descriptor values missing
4. User will select option to remove remaining structures from the molecules list that have missing descriptor values
5. Structures with missing property/activity data are deleted from a molecules list

Extensions:

4a User will substitute missing descriptor values with zero values

4b User will substitute missing descriptor values with mean value of descriptor scale

4c User will remove descriptor scales where variance is too small

Variations:

Postconditions: Molecules list where all structures have descriptor values defined

8.5. Optimise molecular structures

8.5.1. Use case: Convert 2D structure to 3D structure

Scenario: The molecular structure in 2D representation is converted to 3D representation.

Preconditions:
1. Molecules list with structures in 2D representation available
2. Default parameters for 2D to 3D conversion program

Trigger: User selects "2D to 3D conversion" for molecules list

Description:
1. 2D to 3D conversion tool displays default parameters for structure conversion
2. User accepts default parameters
3. 2D to 3D conversion tool shows list of target systems for 2D to 3D structure conversion
4. User selects target system for calculations
5. 2D to 3D conversion tool prepares input for 2D to 3D conversion program, transfers it to the target system and perform conversion on the selected list of structures
6. Generated 3D structures are saved to CDR

Extensions:

Variations:

2a User changes default parameters

4a User selects local system for calculation

4b User selects remote system for calculation

Postconditions: Molecules have their 3D representation with generation info saved to the CDR

8.5.2. **Use case:** Quantum chemical calculation with MOPAC

Scenario: The molecular structure in 3D representation is given as an input for the MOPAC calculation. The resulting output from the MOPAC calculation is saved to the CDR.

Preconditions:
1. Molecules list with structures in the 3D representation is available
2. Default parameters for MOPAC program

Trigger: MOPAC output files are required for the descriptor capculation

Description:
1. MOPAC calculation tool displays default parameters for MOPAC calculation.
2. User accepts the default parameters.
3. MOPAC calculation tool shows list of target systems for QC calculation
4. User selects target system
5. MOPAC calculation tool prepares input for MOPAC program, transfers input files to target system, and performs MOPAC calculations
6. Store reference to MOPAC output files in the storage
7. Replace 3D structure field of molecules with optimised 3D geometries from MOPAC output files

Extensions:

Variations:

2a User changes default parameters

4a User selects local system for calculation

4b User selects remote system for calculation

Postconditions: Molecules in the CDR are updated with references to MOPAC output files and 3D structure field is replaced with optimised MOPAC geometries

8.6. Data warehouse

8.6.1. Use case: **Import structures from DW**

Scenario: User performs query from a selected DW to retrieve a list of structures for further processing in the DM environment.

Preconditions:
1. One or more DWs available
2. Query to match structures (by substructure search, by property, by property value range)

Trigger: User selects "Import from a DW" option

Description:
1. DW import tool presents list of available DWs to user
2. User selects one target DW
3. DW import tools displays the number of structures that matched given query
4. User decides to import matching molecules to the CDR
5. DW import tool retrieves all the structures from selected DW, where given query matched and stores them to the CDR.

Extensions:

Variations:

4a User decides to display all structures

4b User discards result of the query

4c User decides to refine query to be more specific

Postconditions: Molecules stored to CDR

8.6.2. Use case: Select property or activity from a DW

Scenario: User will retrieve a list of available properties from a selected DW and select one property.

Preconditions: One or more accessible DWs

Trigger: Property selection tool

Description:
1. Property selection tool will display list of existing DWs to user.
2. User will select one DW
3. Property selection tool retrieves list of defined properties from a DW
4. User will select one property

Extensions:

Variations:

Postconditions: Selected property

8.6.3. Use case: Retrieve property/activity values from a DW

Scenario: User has list of molecular structures and needs to find given property/activity values for these structures. User will select DW and uses molecules list to perform structure search. Found property values will be saved to molecules list.

Preconditions:
1. Molecules list is selected
2. Property/activity name is defined
3. One (or more) accessible DW

Trigger: User selects "Retrieve property/activity data from DW"

Description:
1. System will show list of existing DWs to user.
2. User will select DW
3. System will perform query in the DW to retrieve property values for given structures. If property values are found then they are saved to molecules list.

Extensions:

Variations:

Postconditions: Molecules list with property values

9. Appendix B: XML Schema Definitions and OpenMolGRID Type Mappings

9.1. http://www.openmolgrid.org/namespaces/StructureList

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
            xmlns=http://www.openmolgrid.org/namespaces/StructureList
           targetNamespace=http://www.openmolgrid.org/namespaces/StructureList
            elementFormDefault="qualified">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      XML schema for a list of structures
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="structureList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="source" type="sourceType" minOccurs="0"
                     maxOccurs="1"/>
        <xsd:element name="structure" type="structureType" minOccurs="0"
                     maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="idType">
    <xsd:restriction base="xsd:long">
      <xsd:minInclusive value="1"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="sourceType">
    <xsd:sequence>
      <xsd:element name="softwareName" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="version" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="method" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="architecture" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="operatingSystem" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="structureType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="coordinates" type="coordinatesType" minOccurs="0"
           maxOccurs="1"/>
      <xsd:element name="enumerationInfo" type="xsd:string"
           minOccurs="0" maxOccurs="1"/>

    </xsd:sequence>
    <xsd:attribute name="id" type="idType" use="required"/>
  </xsd:complexType>

  <xsd:simpleType name="formatType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="CML"/>
      <xsd:enumeration value="MOL"/>
    </xsd:restriction>
  </xsd:simpleType>
```

```
  <xsd:simpleType name="categoryType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="2D"/>
      <xsd:enumeration value="3D"/>
      <xsd:enumeration value="3Dopt"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="coordinatesType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
       <xsd:attribute name="format" type="formatType" use="required"/>
       <xsd:attribute name="category" type="categoryType" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
 </xsd:complexType>

</xsd:schema>
```

Alphabetic list of elements and their corresponding OpenMolGRID types:

Element	*Parent*	*OpenMolGrid Data Type*
coordinates	structure	OMG_STRUCTURE
name	structure	OMG_CHEMICAL_NAME
software	source	OMG_OPTIMIZATION_SOFTWARE
method	source	OMG_OPTIMIZATION_METHOD

Alphabetic list of attributes and their corresponding OpenMolGRID types:

Attribute	*Element*	*OpenMolGrid Data Type*
id	structure	OMG_STRUCTURE_ID
format	coordinates	OMG_STRUCTURE_FORMAT

9.2. http://www.openmolgrid.org/namespaces/SemiempiricalOutput

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
            xmlns=http://www.openmolgrid.org/namespaces/SemiempiricalOutput
       targetNamespace=http://www.openmolgrid.org/namespaces/SemiempiricalOutput
            elementFormDefault="qualified">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      XML schema for the output of semiempirical calculations
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="structureList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="source" type="sourceType" minOccurs="0"
                     maxOccurs="1"/>
        <xsd:element name="structure" type="structureType" minOccurs="0"
                     maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="idType">
    <xsd:restriction base="xsd:long">
      <xsd:minInclusive value="1"/>
    </xsd:restriction>
```

```
  </xsd:simpleType>

  <xsd:complexType name="sourceType">
    <xsd:sequence>
      <xsd:element name="softwareName" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="version" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="method" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="architecture" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="operatingSystem" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="structureType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="coordinates" type="coordinatesType" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="mopacOutput" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="idType" use="required"/>
  </xsd:complexType>

  <xsd:simpleType name="formatType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="CML"/>
      <xsd:enumeration value="MOL"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="coordinatesType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
        <xsd:attribute name="format" type="formatType" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
  </xsd:complexType>

</xsd:schema>
```

Alphabetic list of elements and their corresponding OpenMolGRID types:

Element	***Parent***	***OpenMolGrid Data Type***
coordinates	structure	OMG_STRUCTURE
mopacOutput	structure	OMG_MOPAC_OUTPUT
name	structure	OMG_CHEMICAL_NAME
software	source	OMG_OPTIMIZATION_SOFTWARE
method	source	OMG_OPTIMIZATION_METHOD

Alphabetic list of attributes and their corresponding OpenMolGRID types:

Attribute	***Element***	***OpenMolGrid Data Type***
id	structure	OMG_STRUCTURE_ID
format	coordinates	OMG_STRUCTURE_FORMAT

9.3. http://www.openmolgrid.org/namespaces/DescriptorFile

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
            xmlns=http://www.openmolgrid.org/namespaces/DescriptorFile
            targetNamespace=http://www.openmolgrid.org/namespaces/DescriptorFile
            elementFormDefault="qualified">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      XML schema for a list of structures with descriptor values
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="structureList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="source" type="sourceType" minOccurs="0"
                     maxOccurs="1"/>
        <xsd:element name="metadata" type="metadataType" minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element name="structure" type="structureType" minOccurs="0"
                     maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="idType">
    <xsd:restriction base="xsd:long">
      <xsd:minInclusive value="1"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="sourceType">
    <xsd:sequence>
      <xsd:element name="softwareName" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="version" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="architecture" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="operatingSystem" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="metadataType">
    <xsd:sequence>
      <xsd:element name="descriptor" type="descriptorMetaType" minOccurs="0"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="structureType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="descriptorList" type="descriptorListType" minOccurs="0"
                   maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="idType" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="descriptorMetaType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="type" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="subtype" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
```

```
      <xsd:element name="category" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="idType" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="descriptorListType">
    <xsd:sequence>
      <xsd:element name="descriptor" type="descriptorType" minOccurs="0"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="descriptorType">
    <xsd:complexContent>
      <xsd:restriction base="xsd:anyType">
        <xsd:attribute name="id" type="idType"/ use="required"/>
        <xsd:attribute name="value" type="xsd:float" use="required"//>
        <xsd:attribute name="substSitePosition" type="xsd:integer"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>

</xsd:schema>
```

Alphabetic list of elements and their corresponding OpenMolGRID types:

Element	***Parent***	***OpenMolGrid Data Type***
name	structure	OMG_CHEMICAL_NAME
name	metadata	OMG_DESCRIPTOR_TYPE_NAME
type	metadata	OMG_DESCRIPTOR_TYPE_TYPE
subtype	metadata	OMG_DESCRIPTOR_TYPE_SUBTYPE
category	metadata	OMG_DESCRIPTOR_TYPE_CATEGORY

Alphabetic list of attributes and their corresponding OpenMolGRID types:

Attribute	***Element***	***OpenMolGrid Data Type***
id	structure	OMG_STRUCTURE_ID
	descriptor	OMG_SOURCE_DESCRIPTOR_ID
value	descriptor	OMG_DESCRIPTOR_VALUE

9.4. http://www.openmolgrid.org/namespaces/PropertyFile

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
            xmlns=http://www.openmolgrid.org/namespaces/PropertyFile
            targetNamespace=http://www.openmolgrid.org/namespaces/PropertyFile
            elementFormDefault="qualified">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      XML schema for a list of structures with property/activity values
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="structureList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="metadata" type="metadataType" minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element name="structure" type="structureType" minOccurs="0"
                     maxOccurs="unbounded"/>
```

```
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="idType">
    <xsd:restriction base="xsd:long">
      <xsd:minInclusive value="1"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="metadataType">
    <xsd:sequence>
      <xsd:element name="property" type="propertyMetaType" minOccurs="0"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="structureType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="propertyList" type="propertyListType" minOccurs="0"
                   maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="idType" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="propertyMetaType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="idType" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="propertyListType">
    <xsd:sequence>
      <xsd:element name="property" type="propertyType" minOccurs="0"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="propertyType">
    <xsd:complexContent>
      <xsd:restriction base="xsd:anyType">
        <xsd:attribute name="id" type="idType"/>
        <xsd:attribute name="value" type="xsd:float"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>

</xsd:schema>
```

Alphabetic list of elements and their corresponding OpenMolGRID types:

Element	***Parent***	***OpenMolGrid Data Type***
name	structure	OMG_CHEMICAL_NAME
name	metadata	OMG_PROPERTY_NAME

Alphabetic list of attributes and their corresponding OpenMolGRID types:

Attribute	***Element***	***OpenMolGrid Data Type***
id	structure	OMG_STRUCTURE_ID
	property	OMG_SOURCE_PROPERTY_ID

Attribute	*Element*	*OpenMolGrid Data Type*
value	property	OMG_PROPERTY_VALUE

9.5. http://www.openmolgrid.org/namespaces/ModelBuildingOutput

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
            xmlns=http://www.openmolgrid.org/namespaces/ModelBuildingOutput
       targetNamespace=http://www.openmolgrid.org/namespaces/ModelBuildingOutput
            elementFormDefault="qualified">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Model Building Output Schema.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:complexType name="metadataType">
    <xsd:sequence>
      <xsd:element name="property" type="propertyMetaType" minOccurs="0"
                   maxOccurs="unbounded"/>
      <xsd:element name="descriptor" type="descriptorMetaType" minOccurs="0"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="propertyMetaType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="idType" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="descriptorMetaType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="type" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="subtype" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
      <xsd:element name="category" type="xsd:string" minOccurs="0"
                   maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="idType" use="required"/>
  </xsd:complexType>

  <xsd:element name="modelList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="metadata" type="metadataType" minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element name="model" type="modelType" minOccurs="0"
                     maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="idType">
    <xsd:restriction base="xsd:long">
      <xsd:minInclusive value="1"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="posType">
    <xsd:restriction base="xsd:int">
      <xsd:minInclusive value="1"/>
```

```
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="modelType">
  <xsd:sequence>
    <xsd:element name="type" type="modelTypeType"/>
    <xsd:element name="dependentVariables" type="dependentVariablesType"/>
    <xsd:element name="independentVariables" type="independentVariablesType"/>
    <xsd:element name="R2" type="xsd:double"/>
    <xsd:element name="R2CVOO" type="xsd:double"/>
    <xsd:element name="R2CVMO" type="xsd:double"/>
    <xsd:element name="RMSPEOO" type="xsd:double"/>
    <xsd:element name="RMSPEMO" type="xsd:double"/>
    <xsd:element name="F" type="xsd:double"/>
    <xsd:element name="s" type="xsd:double"/>
    <xsd:element name="trainingSet" type="trainingSetType"/>
    <xsd:element name="legacyData" type="legacyDataType"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="idType" use="required"/>
</xsd:complexType>

<xsd:simpleType name="modelTypeType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="multilinearRegression"/>
    <xsd:enumeration value="partialLeastSquares"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="dependentVariablesType">
  <xsd:choice>
    <xsd:element name="property" type="propertyType"/>
    <xsd:element name="descriptor" type="descriptorType"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="independentVariablesType">
  <xsd:sequence>
    <xsd:element name="descriptor" type="descriptorType"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="propertyType">
  <xsd:sequence>
    <xsd:element name="structure" type="structureType" minOccurs="0"
                 maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="idType" use="required"/>
</xsd:complexType>

<xsd:complexType name="descriptorType">
  <xsd:attribute name="id" type="idType" use="required"/>
  <xsd:attribute name="pos" type="posType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="structureType">
  <xsd:sequence>
    <xsd:element name="observedValue" type="xsd:double" minOccurs="0"
                 maxOccurs="1"/>
    <xsd:element name="predictedValue" type="xsd:double" minOccurs="0"
                 maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="idType" use="required"/>
</xsd:complexType>

<xsd:complexType name="trainingSetType">
```

```
    <xsd:sequence>
      <xsd:element name="size" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="property" type="propertyType" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="legacyDataType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="format" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

</xsd:schema>
```

Alphabetic list of elements and their corresponding OpenMolGRID types:

Element	***Parent***	***OpenMolGrid Data Type***
property	metadata	OMG_PROPERTY_NAME
descriptor	metadata	OMG_DESCRIPTOR_TYPE_NAME
F	model	OMG_MODELSTATISTIC_FISCHER
R2	model	OMG_MODELSTATISTIC_R2
R2CVMO	model	OMG_MODELSTATISTIC_R2CVMO
R2CVOO	model	OMG_MODELSTATISTIC_R2CVOO
RMSPEMO	model	OMG_MODELSTATISTIC_RMSPEMO
RMSPEOO	model	OMG_MODELSTATISTIC_RMSPEOO
s	model	OMG_MODELSTATISTIC_STD_DEV
type	model	OMG_MODELTYPE_NAME

Alphabetic list of attributes and their corresponding OpenMolGRID types:

Attribute	***Element***	***OpenMolGrid Data Type***
id	descriptor	OMG_SOURCE_DESCRIPTOR_ID
	property	OMG_SOURCE_PROPERTY_ID
	structure	OMG_STRUCTURE_ID