Deliverable D2.4

|  |  |
| --- | --- |
| Project Title: | Developing an efficient e-infrastructure, standards and data-flow for metabolomics and its interface to biomedical and life science e-infrastructures in Europe and world-wide |
| Project Acronym: | COSMOS |
| Grant agreement no.: | 312941 |
|  | Research Infrastructures, FP7 Capacities Specific Programme; [INFRA-2011-2.3.2.] “Implementation of common solutions for a cluster of ESFRI infrastructures in the field of "Life sciences" |
| Deliverable title: |  |
| WP No. | 2 |
| Lead Beneficiary: | 11. IPB |
| WP Title | Standards Development |
| Contractual delivery date: | 30 September 2013 |
| Actual delivery date: | 30 September 2013 |
| WP leader: | Neumann 11. IPB |
| Contributing partner(s): | 11. IPB, 1.EBI , 14 UOXF, 12 UB2, 13 UBHam  and others WP participants providing data  Michael Wilson |

*Authors: Daniel Schober, Steffen Neumann, Michael Wilson*

**Contents**

[1 Executive summary 3](#__RefHeading__1454_1605960097)

[2 Project objectives 3](#__RefHeading__1456_1605960097)

[3 Detailed report on the deliverable 4](#__RefHeading__1458_1605960097)

[3.1 Background 4](#__RefHeading__1460_1605960097)

[3.3 Description of Work 4](#__RefHeading__1462_1605960097)

[3.3.1 Requirement analysis and Use case specification 4](#__RefHeading__1464_1605960097)

[3.3.2 Basic overall Design Considerations 6](#__RefHeading__1466_1605960097)

[3.3.4 XSD Development history 10](#__RefHeading__1468_1605960097)

[XSD Issues 10](#__RefHeading__1470_1605960097)

[CV Development History 10](#__RefHeading__1472_1605960097)

[3.4 Next steps 14](#__RefHeading__1474_1605960097)

[4 Publications 15](#__RefHeading__1476_1605960097)

[5 Delivery and schedule 15](#__RefHeading__1478_1605960097)

[6 Adjustments made 15](#__RefHeading__1480_1605960097)

[7 Efforts for this deliverable 15](#__RefHeading__1482_1605960097)

[Appendices 15](#__RefHeading__1484_1605960097)

[Background information 16](#__RefHeading__1486_1605960097)

# 1 Executive summary

The Standards development work package (WP2) here delivers the essential exchange standard for vendor agnostic Nuclear Magnetic Resonance (NMR) spectrum data, with a focus on capturing data that is useful in NMR based metabolomics experiments. After the formulation of UML use case diagrams, we agreed upon design principles (technical and content-wise) and the overall development setup.

We prepared a set of documents to define the format as well as documentation and example files to demonstrate the intended use to our target users. The current versions of these documents were distributed via nmrml.org as release candidates with the goal of generating initial user feedback and to facilitate the integration and development of software tools before the first finalized version is released. The following describes the documents that we have prepared:

● **nmrML.xsd ( nmrml.org/schema/1.0.rc1/nmrml.xsd )**

○ An xml schema that defines the structure, content and semantics of the nmrML documents. The XML schema definition (XSD) is in the XML Schema 1.1 format following the W3C recommendation ( w3.org/XML/Schema ). The schema allows for the capture of raw NMR spectrum data and acquisition parameters for both one-dimensional and two-dimensional spectra, including two-dimensional J-resolved spectra.

● **nmrCV.owl**

○ The controlled vocabulary (CV) describes the more variant terminology in a standardized and unambiguous way. This ontology is the MSI-sanctioned successor of artifacts developed previously at EMBL-EBI, Hinxton, UK (D. Schober, Sansone Group) and the Wishart Research Group, Edmonton, Canada (J. Cruz). This CV currently covers the description of NMR spectrum acquisition set up and raw data generated during the acquisition. There is less coverage of data generated by analysis of the spectrum such as metabolite quantification and identification. The CV terms are used within the nmrML xml file, at positions specified in the xsd, e.g by CVParam references.

● **xml example files**

○ Four XML instances complying with the XSD were generated to illustrate the usage of nmrML in a practical experiment data annotation. These instances also served to test the XSD and CV on coverage, structural soundness and to test parser software.

● **mapping file**

○ SHOULD WE DELIVER THIS ALREADY or for D2.5? To constrain data entry and to verify validness of CV term usage in the nmrML XML files and to be able to enforce minimal metadata standards [Ref].

● **HTML documentation files**

○ Documentation was generated with automated tools that describes the nmrML XSD and the CV OWL and made available via nmrml.org. The documentation allows non-XML and non-ontology savvy end-users open, browse and comment on the standards as well as facilitating the use of the data format by developers and the implementation of tools that use the read or write nmrML. To further ease adoption we also created supplemental documentation and tutorials made available via the same site.

Rudimentary nmrML parsers are available, which read in Bruker or Varian NMR raw data files and generate nmrML schema compliant XML instances.

The parsers are developed in close collaboration with two important open-access NMR data processing tool developers (Batman, rNMR).

The development mood is good and we believe we are in line with the given time scheme and deliverable.

# 2 Project objectives

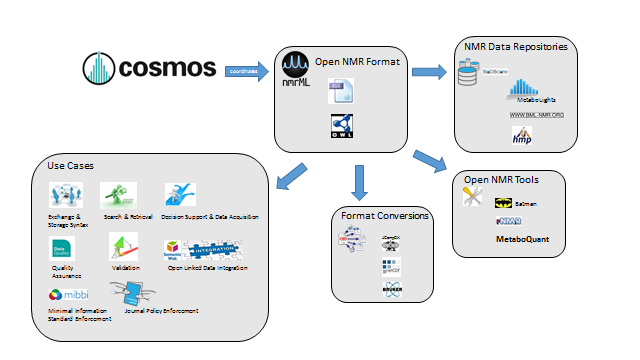
With this deliverable, the project has contributed the following objectives:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Objective** | **Yes** | **No** |
| 1 | Exchange format for metabolomics raw data (XSD) | X |  |
| 2 | Exchange format for metabolomics raw data (CV) | X |  |
| 3 | Example xml files illustrating usage of the standard with example data | X |  |

# 3 Detailed report on the deliverable

## Background

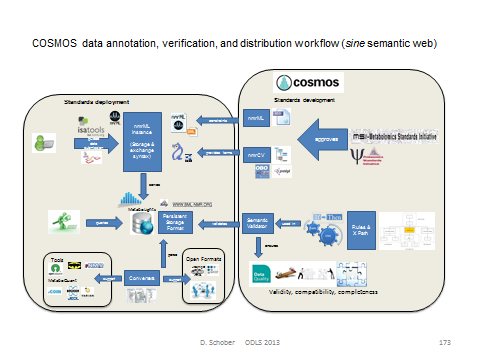
The Proteomics Standards initiative (PSI) has developed a number of XML based data exchange standards, which proved of great usability in proteomics data standardization and intelligent data access. Leveraging on this success the Metabolomics Standards Initiative (MSI) had kicked off an analogous development program with the aim to standardize NMR based metabolomics data. To restart this effort, to leverage and canonize existing predecessor artifacts and to coordinate further developments, the COSMOS EU project was granted. Our aim as COSMOS WP 2, leading the standards development, is to create an open exchange data standard to allow metabolomics data, especially NMR raw data to be shared and stored in an agreed-upon stable and persistent, yet flexible XML format. A bird’s eye view on the envisioned nmrML use cases is provided in Fig. 1.

**Figure 1**: ALL PICS WILL BE PUT IN LATER IN BETTER RESOLUTION.Illustration of data management facilitation by means of a common nmrML standard

## Description of Work

### Requirement analysis and use case specification

The detailed usage of nmrML is outlined in Figure 2.

**Figure 2**: The detailed role of the nmrML xsd and CV within multiple agreed-upon use cases is shown.

#### UML Use case diagram

We developed a UML use case diagram (Fig. 3) to illustrate the distinct usages of nmrML in a standardized manner.

**Figure 3:** UML use case diagram illustrating the usage of the nmrML standard

#### Selecting good example data sets

We defined characteristics of what we believe is a good example data set:

* *The data was gathered in a* p*rototypical, abundant experiment set up, representative for metabolomics data acquisition*
* *The data should stem from a simple experimental set-up (e.g. 1D H+ NMR data)*
* *The data has a published paper available (not a method-, but a research-paper)*
* *The data has a database entry available, e.g. in Metabolights, or HMDB*
* *The data has accompanying original data files (FIDs)*
* *The data is using an abundant vendor format like Bruker or Varian standard files*
* *The data is associated with a responsive contact person, in case someone needs to get back to the data producers to be able to gather additional information or resolve questions*
* *The data has been analyzed with open source tools like Batman or metaboquant*

According to these criteria we have collated example data sets to be converted into nmrML. These example instances can be found in the corresponding github ‘example’ folder, together with an accompanying readme file illustrating its generation or on the documentation page at nmrml.org/schema.

#### Competency Questions for CV development

A set of Competency Questions (CQ) was defined for nmrCV & nmrML. In accordance to functional tests in software development, these CQ serve to evaluate the format for coverage and structural suitability at the later evaluation phase:

MOVE INTO ANNEX ?

* Get spectra generated on a 500Mz Bruker machine in 1D set-up and H+ as acquisition nucleus (on Human Urine samples for doping chemicals).
* Get spectra generated via Bruker CryoProbe and D2O solvent.
* Show experiments generated with samples with a pH range from 6.5 to 8
* Get spectra according to decoupling method.
* Find NMR spectra that have been Fast Fourier Transformed and were smoothed with Gaussian smoothing.
* Find reference spectra for 1-Methylhistidine with a frequency of 600 MHz.
* Did a spectra use flow high resolution probe in the instrument?

CQs for nmrCV and IdentML CV:

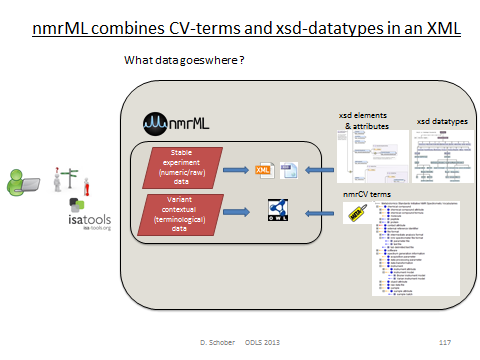
* Find 1D spectra with doublets in ppm range x to y.
* Find NMR spectra for changes in metabolites involved in citrate cycle after fat consumption in human.
* How does the aromatic amino acid fraction differ in Hop plant variants ?
* Find spectra that were generated via open source NMR software.

### Basic overall design considerations

We had several overarching goals that guided our decision making process. The data format should:

* Allow NMR data to be easily shared in a vendor agnostic manor
* Record enough information about an NMR spectrum acquisition to allow for further processing of the raw spectrum without referring to the original vendor files.
* Be easy for developers to understand and integrate into software
* The data format should be flexible and allow for multiple use cases of NMR experiments.
* The data format should be easy for developers to understand and integrate into software.
* The data format should reference the original files for the sake of posterity and in the case where original vendor specific information is required.

As in our PSI role model, we agreed on implementing a combined standard using XML and accompanying CV terms (Fig. 4), as this allows multiple validation levels to be established: XML syntax and structural validity of XML instances (xml element and attribute positions, order and cardinality) are validated by the XML parser against the XML Schema . The mapping files enforce semantic validity by specifying which CV terms are allowed in an element as well as the order and cardinality those terms. A proprietary validator tool (to be developed for the next deliverable) checks that the criteria outlined by the mapping file are being met in a given XML instance. The mapping file combined with the CV can also be used when creating an interface that records NMR experiment information for example to populate a drop down menu or an autocomplete box.

**Figure. 4:** nmrML consists of an XSD specification capturing the more data-near and less variant raw data and a CV, capturing the more variant contextual terminology on NMR.

#### Criteria defining the border between XSD and CV

PUT IN ANNEX?

The XSD branches out into CV-usage, where:

* The terms describe contextual metadata, rather than NMR raw data
* The terms are unstable, variant & dynamically evolving, or need to be changed and updated often
* The terms refer to software names/versions, processing parameters etc.
* The terms are better maintained by a fast reacting NMR user community
* The terms reside at the domains leaf node level
* The terms are search attributes for data querying and DTB-integration
* The terms should be accessible to rule-based reasoning and validation
* The terms should be exploited by profiting from robust subsumption, i.e. exploiting the taxonomic CV backbone

### XSD Development

We started the nmrML.xsd development by modification of the J. Cruz xsd predecessor and under amendment with elements and structures from the BML-NMR XSD developed by Christian Ludwig and Denis Rubtsov in Birmingham. We used the ideas from the past work and publications to guide what information we needed to capture. We also had specific goals about what additional information we wanted to capture in order to meet our overarching goals for the format. In particular we wanted to capture the information about the file creation including original vendor files that were converted to nmrML, software that was used for creation or processing, and contact information for the file creators.

Regarding NMR specific information our aim was to capture all details required for both reproduction of the NMR acquisition as well as further processing of the raw data. In cases where the variability in information to be captured is very high, we rely on the CV to capture relevant information. For when example capturing information about the NMR instrument configuration there are many different possibilities such as probe heads, auto-samplers, brands, models, etc. Another point that that is difficult to capture here is the pulse sequence, which can be completely customized by a user. In this case we opted to capture the names of several of the most common pulse sequences or allowing a reference (via a URI) to the pulse sequence program source code. While not readable in a vendor agnostic manor this decision still allows for most experiments to be easily reproduced while also allowing more custom information to be captured. We decided on which acquisition parameters must be captured through consultation with domain experts, and reference to past work that attempts to define the minimal information required to describe an NMR acquisition.

While vendor formats have multiple different methods of encoding the free induction decay (FID) signal, each with various pros and cons, we decided to standardize the encoding with the goal of making the format easier to read and integrate into software, and following some common practices for encoding binary data in XML. While the binary FID data ends up being the bulk of the bytes in an nmrML instance, the file size remains easily small enough that an nmrML file is easily transferable via web or email. The ‘encodedLength’ attribute in the tag surrounding the FID data allows for software that skips these bytes when reading the file if they are not needed.

The format also allows for the capture of a processed FID and information about the processing of the spectra. It is common to transform the FID from time domain to frequency domain before any further analysis and we felt that this process is so common that it could be considered another form of raw data. Capturing the transformed data makes the nmrML format more practical since time domain data is not usually viewed by users and allows other formats such as IdentML, or QuantML to reference the spectra contained in an nmrML instance.

#### XSD top level structure

An nmrML instance is split up into multiple sections that organize the information in an intuitive way that facilitates easy understanding of the format as well as making development of software application easier. Reference to the CV’s used in the instance are recorded in the ‘cvList’ element at the top, which allows for unambiguous references to CV terms. The ‘fileDescription’ element captures a general description about the file and its contents which allows for easy categorization of different types of nmrML instances eg. 1D vs 2D. The ‘contactList’ element captures information that allows one to contact the original creators of a file in the case that further clarification is needed. The ‘sourceFileList’ contains information about the original files used to make the nmrML instance including files that were required during the acquisition of the spectrum, for example a Varian processing parameter file or a source code file for a pulse program. Similarly the ‘softwareList’ element captures references to software that was used during data acquisition and processing, and may include several different pieces of software. The ‘informationConfigurationList’ element contains information about the configuration of an instrument beyond the acquisition parameters, for example the brand and model of the instrument. The ‘acquisition’ element captures the processing parameters used during the acquisition. Since vendors have their own set of names for each of these parameters, we have standardized them with intuitive clear names. This element also contains the captured FID data. The ‘spectrumList’ element contains 1 or more spectra in the frequency domain.

The current top-level structure of the nmrML XSD is described in Fig. 5.

**Figure 5:** The root near xml elements of the current nmrML.xsd schema, illustrating its main elements. For detailed documentation we refer to the HTML documentation, or the XSD itself, in which extensive element annotations explain the usage of the elements.

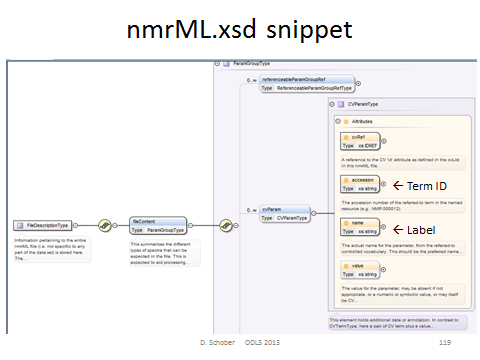
#### CV term referencing mechanism

We here outline how CV term usage is specified in the XSD (See Fig. 6). The requirement for a CV term occurrence in an xml is specified in the XSD by *reference elements* as illustrated in Table 1. Keep in mind that the last element captures free text and makes no CV reference.

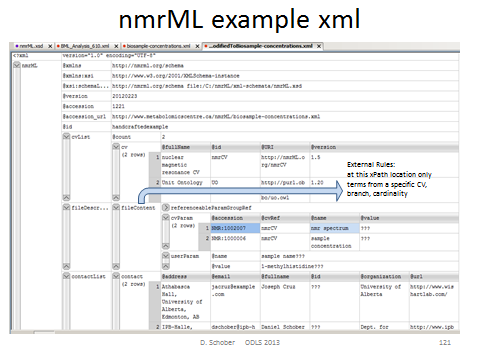
SHOULD WE PUT THIS IN ANNEX ?

**Table 1:**Illustration xml element tapes used for CV and user parameter entry.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reference Element Type** | **Definition** | **Attributes** | **Example** | **Comment** |
| CVTermType | Elements of this type hold annotation as a simple CV term with no further values (Parameters) associated with it. The “CVRef” attubte contains an id unique to the XML instance that is defined in the cvList element. This allows for multiple CVs to be referenced unambiguously. The “accession” attribute contains the ID of the CVterm which is unique within the CV. The “name” attribute contains the term which allows using the term in a program (for example displaying it to a user) without requiring the CV file to be downloaded and parsed. Only controlled CV terms are allowed here. | CVRef, accession, name | DO WE NEED THE LAST TWO COLUMNS ? |  |
| CVParamType | Elements of this type hold additional data or annotation. In contrast to CVTermType, here a CV term plus a value is captured. Only controlled terms are allowed. | CVRef, accession, name, value |  |  |
| CVParamWithUnitType | Elements of this type hold controlled term describing a parameter, as well as a value and a description of the unit the value is recorded in. Only controlled values are allowed here. The ‘unitAccession’, ‘unitName’ and ‘unitCvRef’ attributes are used in the same way to describe the unit as the ‘accession’, ‘name’ and ‘cvRef’ terms are used to describe other CVTerms. The unit ontology is typically used to provide the terms for the unit. | CVRef, accession, name, value, unitCVRef, unitAccession, unitName |  |  |
| ValueWithUnitType | Elements of this type hold a value and a reference to the unit the value is recorded in, but is used in locations where the type of value is already defined by the element, but the unit of the value still needs to be recorded. | Value, unitAccession, unitName, unitCvRef |  |  |
| UserParamType | Elements of this type hold uncontrolled user parameters (essentially allowing free text). For cases where no suitable CV term exists. Before using these, one should verify whether there is an appropriate CV term available, and if so, use the CV term instead. This list can however later be exploited to generate corresponding term requests in given ontologies or CVs. | Name, valueType, value, unitAccession, unitName, unitCvRef |  |  |

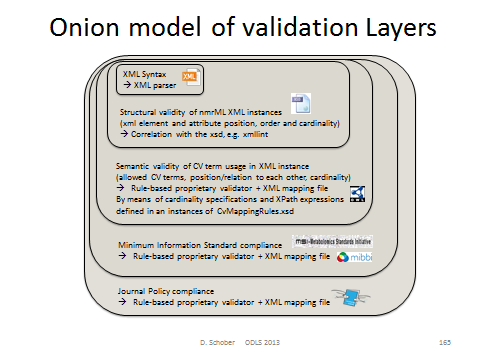
**Figure 6**: Specification of CV term usage via the CVParam element in the XSD

An example of how a CV term is used in an example XML instance can be found in Fig. 7.

**Figure 7**: Example xml instance illustrating instantiation of CV terms to describe a concrete file content via CV Parameters

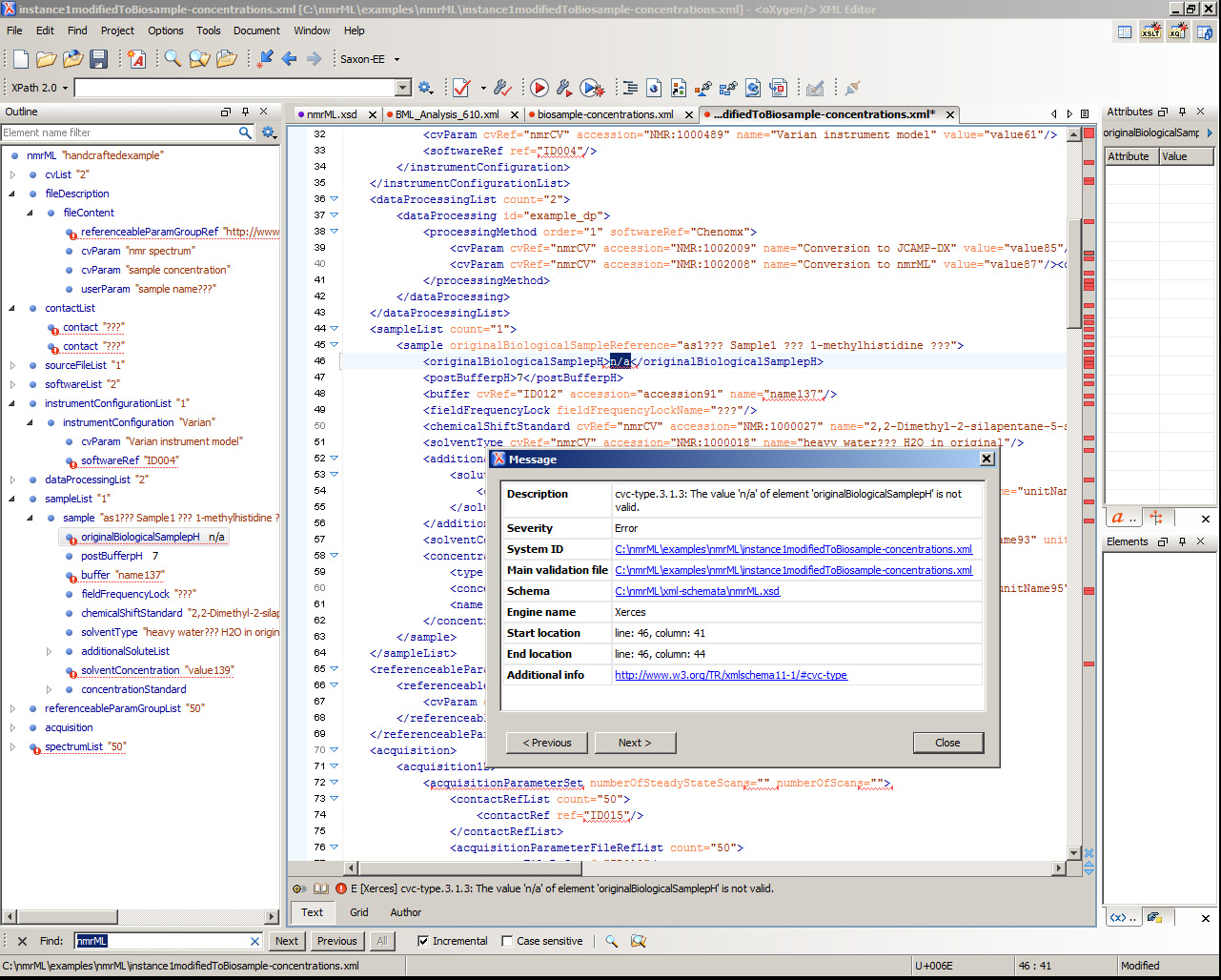
### nmrML Validation approach

Semantic validity of CV terms used in a valid XML instance (allowed CV terms, position/relation to each other and cardinality) is validated by rule-based proprietary validators which exploit xsdToCV mapping files (ready by next deliverable). By means of cardinality specifications and XPath expressions defined in an XML mapping file (an instances of the CvMappingRules.xsd), we define what CV terms are allowed in a specific location of the XSD data model (Fig. 8).

**Figure 8:** Multiple validation layers constrain data entry and support quality assurance

#### An example of syntactic validation of the xml

We see from the Oxygen xerces test on the example xml (homolog to the J.Cruz example) how the XSD already helps constraining the user Input to senseful data, e.g. in the following case for the pH value which is defined to be of type double in the XSD, but was put as String “n/a” in the xml instance, which hence lead to an error message as shown in this GUI screenshot: PUT IN ANNEX ?

**Figure 9**: xerces xml parser error message on wrong data type in xml instance

### CV development history

After agreement on the set up of development tools (Protégé 4), we formulated our CV design principles, namely agreed on file names, format syntax, namespaces, (auto) term ID schemes, a term obsoletion policy, as well as versioning & release procedures. From the given predecessor CVs we proceeded in a bottom-up and middle out Approach to expand the CV. We first added CV terms as required in the XSD leafs, i.e. where CVTermType, CVParamType, CVParamWithUnitType references occure in XSD elements. After this we continued in a use case driven term population. No high throughput term-additions were attempted in our early design phase, as this would clutter CV with terms of doubtful need, impair orientation in the term tree as too many terms distract us from getting the main structure right.

### CV design decisions

#### Choosing a CV exchange syntax

We choose the **OWL Syntax over the OBO format** as exchange syntax for the CV. The reasons were that the OBO tools are instable, the OBO format is only established in the biology domain (lack of off-the-shelf development tools, OBO expressivity not as formal as OWL-DL) and there are hence less resources to integrate with.

#### CV expressivity and semantics

We maintain a pure taxonomy without use of axiomatic definitions. Multiple parenthood is however allowed but needs to be maintained manually, as DL reasoning is not possible without DL axiomatisations.

#### Minimal metadata on a CV term

Representational Unit (RU) metadata is captured via standardized owl annotation properties drawn from imported artefacts like DC, skos and IAO. Not all of our terms currently have natural language definitions as these are time-intensive. None has deeper provenance data explicitly annotated (there is only an implicit indication on from which predecessor CV a term came in the ID ranges). We try to avoid getting stuck in the meta-ether, and had been pragmatic about this.

A term batch submission table should have the following mandatory fields:

term name (rdfs:label)

term definition in natural language (IAO\_0000115, or skos ?)

superclass (ideally a term from the current nmrCV.owl, or an own suggestion)

Optional fields:

synonym (oboInOwl:hasExactSynonym)

term definition source (dc:source)

dc:contributor

dc:creator

example of usage (skos:example)

#### Top Level Ontology usage

There are a few top and upper level ontologies established in the ontology domain. From BFO, OBILight & BioTopLight (btl2), we choosed btl2 as top level ontology to guide our CV upper level development. The reason was that the WP2 leads are involved in btl2 development (fast to react) and btl2 provides a proper set of object properties (close to Relations Ontology). At the moment only a few relations from unit ontology (UO) are used. Bridges from btl2 to BFO & other TLOs exist and we can at some later point still switch the TLO, as we don‘t use any axioms anyway (It‘s only ~10 classes anyway, rebinning is easy). It can be argued why we use a TLO when developing a CV not an Ontology. There has already been a case where the TLO provided modeling restrictions that allowed an automatic DL reasoner to discover CV modelling errors, e.g.<https://github.com/nmrML/nmrML/issues/62>

Nevertheless, at the moment we avoid any usage of object properties from the CV. E.g. for a software vendors file format, we could have in the CV:

NMR Instrument hasVendor Vendor

Instead, we say in the mapping file that for an Instrument, the Name and Vendor has to be specified.

#### Term reference and import mechanism

There are four possible ways to reuse existing CV terms from other ontologies. We majorily used the first method:

1. use the terms in the CV by ID reference (e.g. as done with IAO metadata). This option is fast and flexible, but no metadata on used terms available.

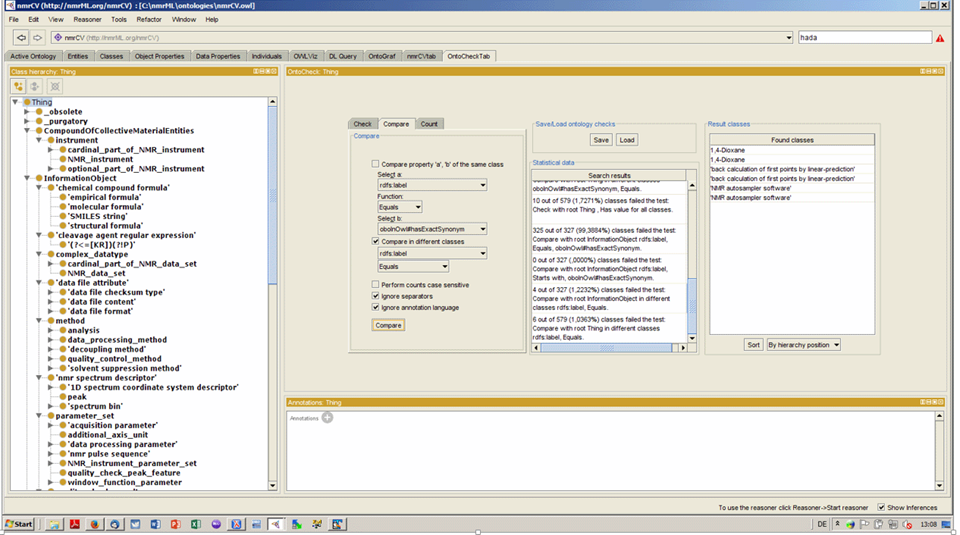
2. use the MIREOT term referencing method. This option is too complicated and relies on outdated scripts

3. use full owl:import statements (e.g. as done for UO). This option however clutters the CV with seldomly used terms, occupies RAM, but retains all metadata. This option is overshot for most use cases.

4. use dbxref statements. These are easy but not a standard way in OWL (these annotation properties are provided by the OBOinOWL namespace)

#### Term naming conventions

We apply a labelling scheme in accordance to <http://www.obofoundry.org/wiki/index.php/FP_012_naming_conventions>. The OntoCheck P.4 plugin (Fig. 10) is used to avoid term redundancies, i.e. to check on redundant labels, e.g. it detected that ‘TecMag’ was included twice, once under http://nmrML.org/nmrCV#NMR\_400285 (NMR data format) and once under http://nmrML.org/nmrCV#NMR:1400255 (NMR\_vendor). This redundancy could then be removed by specifying a more explicit label.

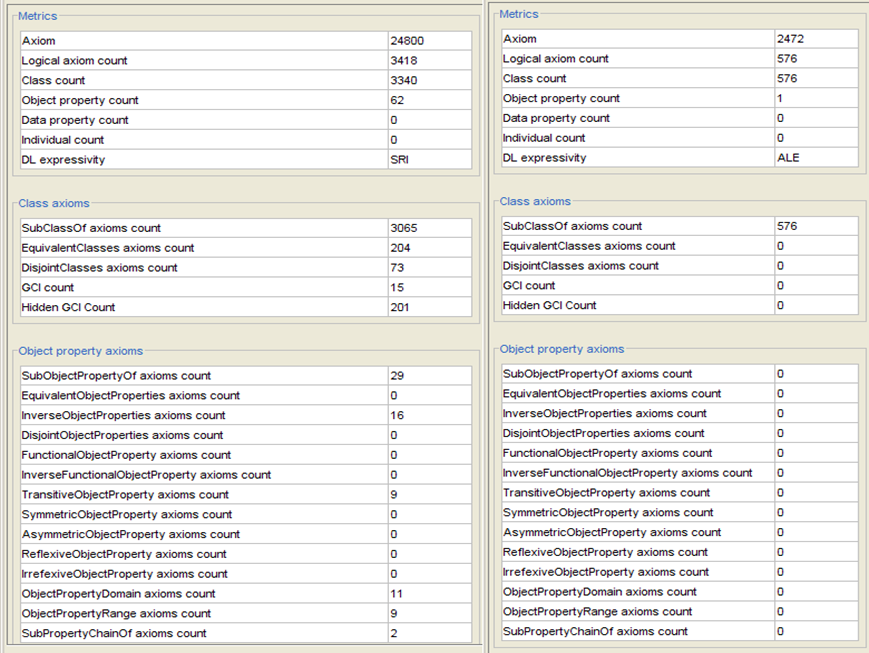


**Figure 10:** A screenshot displaying maintenance of the CV in the ontology editor Protégé 4. The OntoCheck Tab is shown which displays the CV term hierarchy to the left and allows to specify and label comparison check to discover redundant labels.

### XSD current status and metrics

[Michael: todo]

### CV current status and metrics

We here provide the statistics describing the CV (Figure 11).

**Figure 11**: The left side displays the metrics of the nmrML-CV with, the right side without the imported ontologies (btl2 and UO).

### Source files and documentation

All source files are available on the project Github pages, together with an accompanying readme file. A first prototype version of the nmrML XSD, accompanying CV and example XML instances are available under the Github development pages.

**The nmrML XML Schema (XSD):**

<https://github.com/nmrML/nmrML/blob/master/xml-schemata/nmrML.xsd>

**The nmrML Controlled Vocabulary (CV):**

<https://github.com/nmrML/nmrML/blob/master/ontologies/nmrCV.owl>

**The nmrML example files:**

<https://github.com/nmrML/nmrML/tree/master/examples/working.tmp/nmrML> &

<https://github.com/nmrML/nmrML/tree/master/examples>

Browsable HTML serializations of the XSD and the CV can be found in the github folders nmrML\docs\SchemaDocumentation\HTML\_Serialisations and nmrML\docs\CVDocumentation\OwlDoc respectively.

**GitHub:** <https://github.com/nmrML/nmrML>**Cosmos website:** [http://www.cosmos-fp7.eu](http://www.cosmos-fp7.eu/)

**nmrML website:** [http://nmrml.org](http://nmrml.org/)**nmrML wiki:** <http://cosmos-fp7.eu/nmrML/>

**nmrML google forum:** [https://groups.google.com/forum/#!forum/nmrml](https://groups.google.com/forum/)

### Example implementations

We created two example xml files from our use cases to serve as data-driven check on the format.

**Example 1:**

At first, we analyzed, if our schema compensated for all data required by the original predecessor. The original J Cruz nmrML XML example was taken from <http://www.metabolomicscentre.ca/nmrML/biosample-concentrations.xml> and was transliterated into an nmrML XML instance generated via Oxygen as described at <http://www.oxygenxml.com/doc/ug-editor/topics/xml-schema-instance-generator.html>Where the correct entity usage for some values were doubtful, value entries were marked with the String "???". Unused elements and attributes containing the mere default autogenerated values were deleted in the final version.

**Example 2:**

An example was created from a reference spectrum obtained from HMDB (<http://www.hmdb.ca/spectra/nmr_one_d/1024>). The file was initially written manually by hand, obtaining values to fill in the file from the Varian procpar file and a python script for encoding the raw FID data into the correct format. This example also proved useful for creating the conversion software since the output could be compared.

After we developed the conversion software, more example files were generated from MetaboLights entries MTBLS1 and 25 data as well as for IPB Hop data [Ref].

## Next steps

The next step is to plan the first release of the core XSD and initial CV. Further testing of the XSD is required with diverse experimental configurations, to ensure that our goal of flexibility has been achieved. We must also ensure that the schema is compatible with the steps we are taking toward QuantML and IdentML. Continuing to improve the documentation and building a community of users will provide further feedback for improvements to the Schema. At the same time we will continue the data-driven CV expansions and add new terms according to the additional examples selected by our different partners. On the CV side we also need to integrate new EBI-NMR CV classes (using tabular mass term import).

In general we have to extend the format specification, e.g. adding more experimental metadata, such as sample types as well as more information on metabolite identification and quantification (both XSD and CV side). Also we need to work out an evaluation pipeline. As part of the next deliverable (D2.5 - Real data, Converters, Validators and Parsers for NMR-ML, m24), we will implement the CV-aware validator software and mapping files containing verification rules in order to check xml instances on semantic errors and completeness. In parallel we are implementing parsers for format conversions and I/O to open source tools. The creation of ISA Tab specifications for easy tabular data entry and minimal reporting requirement enforcement is considered a further next step (D2.6).

# 4 Publications

Schober D., Mayer G., Moing A., Eisenacher M., Neumann S., **Ontological analysis of controlled vocabularies used in PSI/MSI supported XML standards**, Workshop: ODLS 2013, GI-Edition Lecture Notes in Informatics, *Proceedings of the Jahrestagung der Gesellschaft für Informatik 2013*, Matthias Horbach (Hrsg.), Koblenz, Germany, 16.–20. September 2013, p. 1875-1888,<https://wiki.imise.uni-leipzig.de/Gruppen/OBML/Workshops/2013-ODLS-en>

Metabolomics letter/paper ?

# 5 Delivery and schedule

The delivery is delayed: ◻ Yes ☑ No

# 6 Adjustments made

None.

# 7 Efforts for this deliverable

|  |  |  |  |
| --- | --- | --- | --- |
| **Institute** | **Person-months (PM)** |  | **Period** |
|  | **actual** | **estimated** |  |
| 11. IPB | 3 | 3 | 6 |
| Total |  |  |  |

# Appendices

**Detailed version history of the CV**

* v.1 initial result from the Obo Edit OBO to OWL conversion
* v.2 added RA Metadata (just using standard annotation properties, i.e. DC)
* v.3 added BFO 1.1 import (better for OBO backwards compatibility)
* v.4 This version as v.3, but importing BFO 2.0 instead of non-DL BFO 1.1. BFO 2.0 is experimental, but has a rich set of relations integrated from RO, For BF0 2.0, see http://ncorwiki.buffalo.edu/index.php/Basic\_Formal\_Ontology\_2.0:\_Tutorial\_at\_ICBO/FOIS, file loads from http://bfo.googlecode.com/svn/releases/2012-11-15-bugfix/owl-group/bfo.owl
* v.5 This version as v.4, but additionally importing MSI NMR.owl developed at EBI
* v.6 This version as v.5, but importing BiotopLight2.0 instead of BFO 2.0 as top level ontology
* v.7 This version is a complete new start (as v.6 ended up being too complex and error prone). For this version I removed the unit import from the Wishart nmr.obo, converted it into owl and imported biotop light 2 and the msi-nmr.owl. To make editing easier, I will merge the owl files physically rather than importing the msi-nmr.owl. The tol level classes from OBI and BFO will then vanish as well.
* v.8 This version as v.7, but namespace set to NMR, added \_purgatory helperclass and started rebinning under BiotopLight 2.
* v.9 This version as v.8, but Wishart CV binned under biotopLight2 (btl2). Added RA metadata.
* v1.0 As v.9, but removed OBI temporary and outdated IDs and Refs.Taxonomic re-binning of classes that part\_of /is\_a 'Metabolomics Standards Initiative NMR Spectrometry Vocabularies' under appropriate Biotop classes. Integration of required xsd leaf nodes into CV (see below). Removed Wishart Top Level nodes of doubtful justification, i.e. 'Metabolomics Standards Initiative NMR Spectrometry Vocabularies' and 'spectrum generation information' and 'spectrum interpretation'.
* v1.1 Merged msi namespace nmr ontology (Schober NMR) into Wishart CV (using P4 Refactoring/Merge) in order to get rid of import statements and restriction overriding.
* v1.2 Entity (ID) renaming of newly (physically) integrated MSI NMR Terms from MSI namespace to Cosmos nmrML namespace.
* v1.3 File renaming to get rid of version in Filname (now stores as RA annotation property) infile. New Namespace (now set to http://nmrML.org/nmrCV to distinguish it from xsd namespace). Alignment of ID schemes:To archieve this, we substituted 541 occurances of "nmrCV\_" for "nmrCV#NMR:" in the complete owl file. Then we substituited 710 occurrances of "nmrCV#MSI\_" with "nmrCV#NMR:1" to align the old MSI IDs to the new NMR prefix and 7 digit length. Importing DOAP, added RA metadata using http://usefulinc.com/ns/doap#, then removed doap import to get rid of confusing class top level.
* v1.4 Empty outdated namespace declarations and NS prefix declarations were removed from the file. The following object properties were taken out of the owl file:

http://nmrML.org/nmrCV#has\_regexp

http://nmrML.org/nmrCV#has\_units

http://nmrML.org/nmrCV#part\_of

Their usage in the old Cruz obo file was minor and has to be recreated by hand, but ideally with relations from btl2.

* v1.5 Major restructuring and redundancy removal, i.e. instruments are now captured as instrument attribute/models.
* v1.6 CV is now also covering the term-needs for the BML-NMR XSD. But, again, the CV is still considered to be a prototype. Its coverage can be very shallow at times. For some cases there is merely a corresponding CV Entry Class available (to be referenceable by the xsd), which has no further subclasses. These leaf nodes will have to be expanded successively via our use cases and later by term-requests from the practitioners/users. We can expect the CV to grow from currently to about 2500 Terms (as in PSI MS CV). Labels were aligned to be consistent, i.e. NMR\_spectrum\_post-processing\_parameter\_set was changed to NMR\_data\_post-processing\_parameter\_set to be in harmony with the existing NMR\_data\_pre-processing\_parameter\_set. 'run attribute' was moved into purgatory. Use acquisition parameter instead. This version imports the owl versions of Unit Ontology and PATO (Qualities).
* v1.7 Stop any notion of pre and post-processing (there is no agreement on meaning and start/end). We now use 'frequence domain processing' and 'time domain processing' as sortals for processing parameters.

**CV Issues**

•Reduplication in disjoint trees, e.g. software names or Vendors in file formats, …

•Classifying spectrum prosessing methods according to pre/post acqu, or pre/post FT ?

–Agreed on Time domain Processing and Frequency domain processing

•Border nmrCV (raw data) to nmrIdentCV ?

– Both in one & the same nmrCV file, but could add slim indicator for that.

# Background information

|  |
| --- |
| UPDATE WITH WP INFO  This deliverable relates to WPX; background information on this WP as originally indicated in the description of work (DoW) is included below.  WPX Title: Insert WP title  Lead: Insert WPL name, affiliation  Participants: Insert partners involved in WP  Insert WP summary |

|  |  |  |  |
| --- | --- | --- | --- |
| **Work package number** | WPX | **Start date or starting event:** | month X |

|  |  |
| --- | --- |
| **Work package title** | Insert WP title |
| **Activity Type** | Insert activity type (e.g. RTD, MGT) |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Participant number** | No: partner | 11. IPB | No: partner | No: partner | No: partner | No: partner | No: partner | No: partner |
| **Person months per participant** | XX |  | XX | XX | XX | XX | XX | XX |

|  |
| --- |
| **Objectives**     1. Insert objective 1 2. Insert objective 2 |
| **Description of work and role of participants**  Insert WP description, tasks etc. |
| **Deliverables** |

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **Due**  **month** |
| DX.X | Insert deliverable title | X |
| DX.X | Insert deliverable title | X |