EDIT WP5 Workshop-Report

The new concept editor software - Requirements from user of the Berlin Model

February 21-22, 2008

Botanic Garden and Botanical Museum Berlin-Dahlem

Participants: Walter Berendsohn, Pepe Ciardelli, Markus Döring, Anton Güntsch, Ralf Hand, Andreas Kohlbecker, Henning Kusber, Karol Marhold, Andreas Müller, Eckhard Raab-Straube

1 Purpose of the workshop

The workshop brought together users and developers of the existing Berlin-Model web editor to discuss and compile requirements for the new generation of concept enabled editor software for the EDIT CDM (common data model). As all participating taxonomists were botanists it is planned to schedule a comparable workshop with a broader audience in the near future.
A wiki site was set up prior to the workshop to put together questions and issues to be addressed by the workshop (http://dev.e-taxonomy.eu/trac/wiki/ConceptWorkshop). The site will be kept open and can be used for further comments and discussions.

2 Beyond forms – the new generation of editing software

Pepe Ciardelli gave a presentation of the new Eclipse based editor software. Although the system is still in an experimental state, the presentation made clear that the technology goes far beyond existing form-based data entry systems. The present prototype shows information for single taxa in a text-based window. Navigation between taxa is possible using both a search window and a tree for browsing. Data are stored to the connected database using a save button. It was agreed that

- The taxonomic tree should always highlight the currently focussed taxon and open to reveal all its included taxa.
- The system should allow for saving erroneous names. The only requirement is that a taxon with an empty name string can not be saved.
- Incorrect names can be entered as orthographic variants.
- There should be an attribute qualifying the kind of orthographic variant (e.g. “as published”, “corrected in accordance with ICBN [Tokyo] Article 44.1”).

As the taxon editor (taxon-view) works essential as a text editor, the difference between correcting an existing object (e.g. author name) and inserting a new one can not determined by the system. When a user edits a data object - such as an author - that is shared by others, a new author will be created for the current object, and a warning will be shown that other objects are associated with this author. The user can then choose to apply the new author to all objects, using either a right-click context menu or by choosing from a list of suggestions triggered by clicking on the warning icon.

If re-usable objects (e.g. names) are modified users might want to have feedback on the consequences of their operations. This could be achieved by

- Immediately displaying all affected objects.
- Having a special symbol in cases when a save affects data that are not displayed.
- Having a test button in addition to the save button.

Catalogues (e.g. author lists) should include a possibility to flag standard items. Non-standard items should be deleted when orphaned.

3 Workflow and concept processing – basic considerations

Versioning

It was agreed that versioning should be implemented so that a version contains a snapshot of all data belonging to a taxonomic view at a given point in time. Once a version has been created, modifying it is not allowed even if it contains erroneous data. Otherwise references to it would be invalidated. “Disclaimers” can be assigned to individual taxa.

Versions are explicitly published as a result of a decision process.

In addition to “frozen” versions, a “live-version” can be published which contains the latest state of the dataset (with a caveat “may change on-line”). The correctness of references to this live-version is the responsibility of the system creating the reference.

Within one version multiple different taxonomic opinions for the same taxa can be represented.

Storing facts / features

Both the Berlin Model as well as the Common Data Model can represent facts (“features” in the CDM) linked to the (potential) taxa they were originally associated with. However, existing projects (such as Euro+Med) do not fully make use of the possibility and prefer linking the data to the accepted taxa of their own treatment. It was agreed that

- The new editor should offer the possibility to link facts/features to the potential taxa representing their sources.
- Not every fact/feature should create a potential taxon. It only makes sense for references that give a practical circumscription of the taxon concept (e.g. Floras, Faunas, Monographs and perhaps checklists). Therefore, a view showing all “unassociated facts/features” should be made available.
- Accumulation of facts/features should be possible depending on their type (see. article about “applicability” in MoReTax publication).

Displaying the concept graph

The present Berlin model editor has a rather limited display of the set of concepts contained in a view and their concept relations. This is because it displays only the presently focussed accepted taxon together with all objects directly linked to it. It was agreed that a more flexible view would be helpful which allows for displaying larger subsets of the potential taxon graph. There are existing libraries for displaying graph structures, which could be used for this purpose. It was also agreed that this feature will not have to be integrated into the editor and could be part of the portal software instead.

Combined concept relations

The Berlin Model offers five basic concept relation types and all combinations of them as well as a doubtful flag, which gives a total of 64 possible relationship types between two taxa. Although existing applications haven't used the combined relationships so far, it was agreed that they should be kept and supported by the new editor. The possibility of using them will probably motivate users to create more complex relationships.

Bundling identical concepts

There should be a way to group names from different treatments as identical concepts. This could be a box representing the concept. Adding a taxon to this concept could then be a simple drag and drop operation. More generally, dropping a taxon from the recently viewed or the search window into the taxon window could open a selection for the relationship type.

4 Specific taxonomic operations

Operations used for changing the status of taxa are notoriously complex. The participants agreed on the following strategies to support such operations:

Status change from synonym to accepted

The system should offer to turn the previously accepted taxon into a synonym. This could be implemented as a right-click menu with two options: 1) Make into new taxon. 2) Make it accepted name for the present taxon.

Status change from accepted to synonym

All data associated with the previously accepted taxon should be moved in one step but everything can be sorted by positioning the two taxa next to one another then dragging elements from one to the other:

- Synonyms become synonyms of the new accepted name
- Misapplied names become misapplied names of the new accepted taxon
- Included taxa are moved as included accepted taxa into the new accepted taxon.
- They can be selected and moved collectively into the taxon's synonymy.
- There is an option in the editor's preference section determing if a marker is set indicating that something has been moved. All such markers can be removed in one step.

Move a species from a genus to another one (or infraspecies to another species)

This operation affects the name of the taxon moved as well as the names of its subtaxa. The system should support this operation by suggesting names that are already in the database. New names should not be created automatically.

Any included taxa whose inclusion violates taxonomic naming, i.e. genus does not correspond to parent taxon, will be marked in the taxonomic tree with warnings and corresponding suggestions.

Implicit behaviour

It was discussed how much taxonomic integrity should be enforced by the system based on a number of examples given on the workshop wiki site (see http://dev.e-taxonomy.eu/trac/wiki/ConceptWorkshop). It was agreed that

1. Dragging a name on a homotypic group changes the name relationship for all uses of that name. Other users of that name should be alerted to the discrepancy and need to be able to find out according to who was responsible the name relationship.
2. Basionym author teams are stored with the basionym, not with the combination (if there is none, a dummy basionym is created). A change in the parenthetical author in a combination should not affect the basionym author? But the conflict should be highlighted.
3. Comparably, the zoological name includes the year, so this should be resolved similarly.

In general, participants favoured the highlighting of problems over automatic resolving or strict enforcement of integrity rules.

5 Rights and roles

It was agreed that the new system needs a rights management so that Editors can tightly control access to the content. The rights management should support three principle access types:

- Read only (general, without restrictions)
- Annotate (general, without restrictions, right to change your own annotations)
- Edit (does not differentiate between insert, change, delete)

It was further agreed that the rights management should be able to refer to

- Taxa (e.g. a genus, a family): Edit or not, always annotate
  - Specific taxa within the tree
  - Multiple taxa but not a specific one (doubtfully needed)
  - Possibility to edit related taxa in different treatments (there is problem of assigning rights which has to be further analyzed)
  - Inclusion of inexistent misapplied name:
    - existing concept: link but do not change
    - existing name: create concept but do not change name
    - inexistent name: create name and concept
- Specific object types (names, references, specific features types)
  - Features: yes
  - Specimens: probably yes
  - Distributions: probably yes
  - Authority files: this is a general editor task, so no.
  - Names: this is a general editor task, so no.

Assigning rights based on content (e.g. rights to modify taxa occurring in a specific region for regional editors) is not needed. These operations are sufficiently covered with annotations.

Editing shared objects (e.g. a name used in multiple treatments) is problematic. A possible solution would be to create a new version for every change of the object. Other objects referring to this object would then stay with the old version but they would also be alerted to the fact that a new version exists (with a red wavy underline for example).

6 Miscellaneous

Duplicate records

If a user has to pick from list of duplicate records (e.g. a list of identical author strings), the decision to pick the right one has to be supported with additional information. This could be achieved with a mouse-over window displaying the needed information after 3 seconds for example. (Incidentally, this delayed mouse-over effect could be used elsewhere, for example to show a taxon's synonymy when hovering over a taxon tree node.)

There will also have to be tools for resolving duplications (standardisation tools). It was agreed that these tools would have a different spreadsheet-like user interface allowing for sorting on different fields. For this we will have to identify objects which are typically subject to duplicate resolvements (e.g. names, authors, references) and for each of them identify the set of attributes needed to make an informed decision when merging objects. A diff-tool for identifying differences between selected objects would also be helpful.

Handling types and specimens

A functional and successful editor model for types and specimens can be found in the AlgaTerra software. WP6 will make significant use of this in defining requirements.

Integrity checker

The editor should have an integrity checker identifying problems in the data according to a defined set of taxonomic integrity rules. This could take the form of a user-triggered report. It might be useful to have also a separate report generation in particular for performing test imports.

Maintaining state between sessions

Upon closing the application, the editor should save the state of the user's data locally (not to the CDM, unless explicitly asked to do so). When the user re-opens the editor, all his work should appear in the state in which he left it.