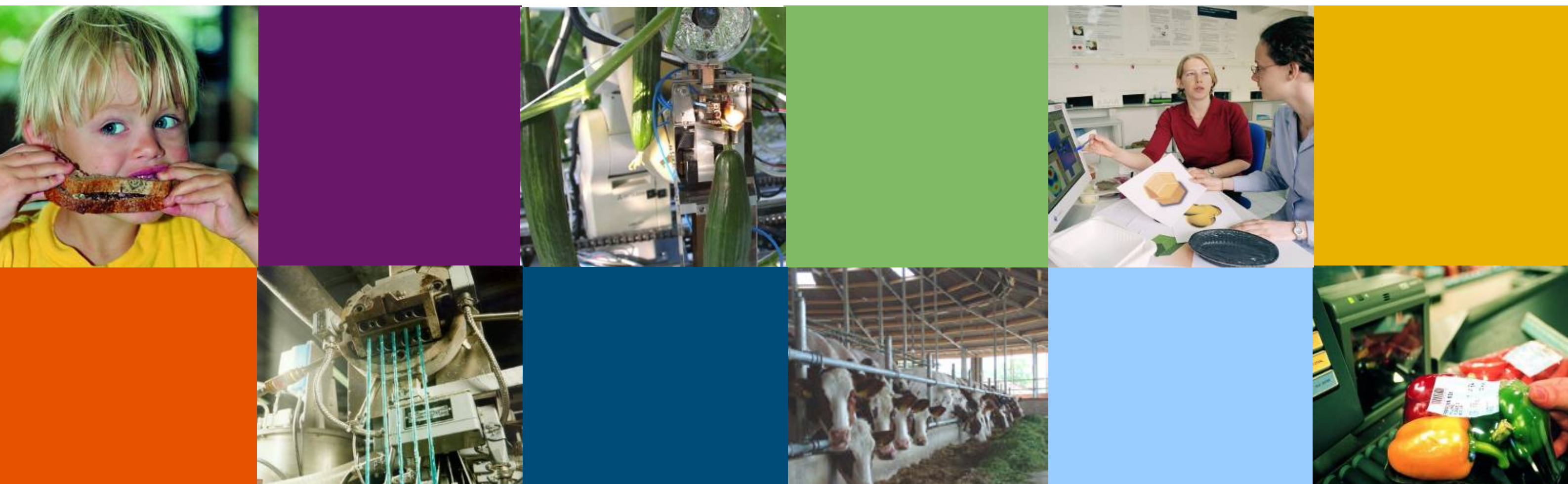
Plan for collaborative working with ROC+

Deliverable Q2 2015

eFoodLab WP1

Jan Vogels
Nick van Hamersveld

Report 1588



COMMIT/

Colophon

Title	Plan for collaborative working with ROC+
Author(s)	Jan Vogels, Nick van Hamersveld
Number	1588
Date of publication	01-08-2015
Confidentiality	No
Approved by	Jan Top

Wageningen UR Food & Biobased Research
P.O. Box 17
NL-6700 AA Wageningen
Tel: +31 (0)317 480 084
E-mail: info.fbr@wur.nl
Internet: www.wur.nl

1 Introduction

Ontologies - structured vocabularies of important domain terms - form the core of intelligent semantic software, such as smart search. In order to facilitate rapid development of ontologies, we have developed the ROC+ application within the COMMIT/ eFoodLab project. This web application allows users to enter important terms, see suggested terms from existing references, and organise the terms in a structure.

Building an ontology is usually a collaborative action between multiple experts working in the same domain. However, ROC+ is currently most suitable for one single user per ontology or project, because simultaneous actions of multiple users can get in each other's way. In particular the Taxonomy step in the ROC+ process, in which items are visually dragged and dropped in a hierarchy, is very sensitive to users performing actions at the same time. For example, a user may put a lot of work into placing several different types of fruit under the concept 'fruit', only to find that someone has deleted 'fruit' in the meantime, and so their work is lost.

We would like to make it possible for multiple experts to work on the same ontology or project at the same time, without their actions interfering with each other. We foresee the following issues which must be addressed to allow intensive collaboration:

1.1.1 Interruption of operations (Unit of work)

Complex operations often consist of multiple smaller actions. For example, in the removal of a concept from the taxonomy, not only must the concept be deleted, but underlying concepts also need to be moved to the parent node of the deleted concept, so they won't be lost. We don't want any interruption from other users during this complex operation, often called a Unit of Work. Interruption could cause data corruption, for example, if a problem occurs halfway through moving the sub-concepts. The concept could, for example, try to be linked to itself, which is an invalid action and as such will be refused. The concept will therefore end up parentless. In such a case the whole operation should succeed or fail, but not partially succeed.

1.1.2 Data out-of-date (Consistency check)

During collaborative working, it can happen that different users want to carry out the same action at the same time. While user A is looking at the taxonomy, user B can remove concepts. A does not see this happening. When A then selects the concept and tries to add items to it, it only exists in the cache on the webpage, and no longer in the database. Therefore, upon e.g. the selection of a concept, the consistency (e.g. the existence of the concept) needs to be checked. If the consistency check fails, then the action must not be carried out.

Handling these aspects will produce an application which can handle multiple users without spoiling the data. However, users would be unaware as to why their action had not been carried out, and would also be taken by surprise by changes made by others. For this reason, it is also necessary to develop appropriate techniques for user notification of errors and of actions made by others.

2 Interruption of operations (Unit of Work)

To avoid interruption of complex operations, such as creating a relationship between two concepts in the taxonomy, we need to lock the concept. Locking is an abstract mechanism to prevent editing by other actions. As long as we are still carrying out sub-actions, the concept is locked, and afterwards (regardless of success or failure) the concept is freed for other actions. This prevents the Unit of Work being disturbed.

3 Consistency check

To go into detail on the consistency check, the following actions can be named for which an interaction between users is possible, and during which errors can occur. For each of these actions, we need to perform a consistency check.
Table 1 shows these actions, organised around the steps of the ROC+ process. Per user action, the checks that should be carried out on the server are given, along with any actions required to ensure consistency. If the action does not pass the check, then the whole Unit of Work must be revoked.

Table 1: Actions and the necessary consistency checks

Action	Check (and actions)
<u>Project</u> ***ROC+ allows users to organise their work in projects***	
Removal of the project	• *Does the project still exist?*
<u>Concept</u> ***Users can add, edit and remove concepts (terms which are important in their domain)***	
Addition of a concept	• *Does the concept already exist?*
Editing the name of a concept	• *Does the concept still exist?* • *Does the new name already exist?*
Removing a concept	• *Does the concept still exist?*
<u>Suggested concepts</u> ***ROC+ searches external sources and suggests concepts which may be interesting for the user***	
Accepting suggested concepts	• *Does the suggested concept still exist?* • *Has the concept already been added?*
Removing suggested concepts	• *Does the suggested concept still exist?*
<u>Synonyms</u> ***The user can indicate that a concept has more than one possible name – synonyms***	
Adding a synonym	• *Does the concept still exist?* • *Does the synonym already exist?*
Add an existing concept as a synonym	• *Does the concept still exist?* • *Does the synonym already exist?* • *Does the concept to which the synonym is being added still exist?*
Removing a synonym	• *Does the synonym still exist?* • *Does the concept still exist?*
<u>Relationships</u> ***The user can indicate that a concept is***	

related to another concept	
Adding a relationship	• *Does the concept still exist?* • *Does the related concept still exist?* • *Does the concept already have another relationship with the related concept?*
Removing a relationship	• *Does the concept still exist?* • *Does the related concept still exist?*
<u>Taxonomy</u> ***The user can organise the concepts in a hierarchy***	
Adding a taxonomy relation	• *Does the concept being linked still exist?* • *Is the concept being linked not locked?* • *<Action: Lock the concept being linked>* • *Does the 'target' concept still exist?* • *Is the target concept not locked?* • *<Action: Lock the target concept>*
Changing a taxonomy relation	• *Does the concept being linked still exist?* • *Is the concept being linked not locked?* • *<Action: Lock the concept being linked>* • *Does the 'target' concept still exist?* • *Is the target concept not locked?* • *<Action: Lock the target concept>*
Removing a taxonomy relation	• *Do both concepts still exist?* • *Is either concept locked?*

4 Validation and error handling

In order to help the user to understand what is happening during collaborative working, we need to give clear feedback during errors. This means that we need to pay attention to validation and error handling.

4.1.1 Validation

Validation handles errors made by the user, when the user has tried to carry out an illogical action. Actions initiated by the user can be validated by logic. For example, a concept being linked to itself is not allowed, so this action must be caught during validation, and prevented.

4.1.2 Error handling

We can define two types of error handling: expected and unexpected error handling. For expected scenarios we can show specific error messages as feedback, for example 'Sorry, the concept you are trying to edit no longer exists'. For unexpected errors we can only report that something unexpected has gone wrong.

Both validation and error handling produce messages, which will be displayed to the user in a clear manner when an error occurs.

5 Information about changes made by others

It is necessary to be aware of activities of fellow users in the same ontology, so as not to frustrate each other. Now that we are going to support multiple users working in the same ontology, some indication of the activities of others must be integrated into the user interface. During some actions, namely 'drag-and-drop', concepts will be temporarily locked by other users. This can be indicated visually in real-time to the user, for example by highlighting the concept in a different colour, or showing a lock icon.

This indicates that an item is in use, but does not give any information as to the actual actions causing the lock. Possible options for indicating actions are

- Real-time visual updates
- Pushing data
- An activity log

5.1 Real-time updates

If we implemented real-time updates, then the user could see the other users dragging and dropping items in real-time on their screen. This would provide very direct information about the actions of other users. However it appears undesirable to us for the screen to be continually refreshed while the user is working, because this can be very disturbing. For example, if you choose a concept in a taxonomy to add a concept to, then you could see the concept being dragged away by someone else. This is annoying, potentially disorientating, and also unnecessary, as adding a concept is technically possible, even if the target concept is being moved.

5.2 Pushing data

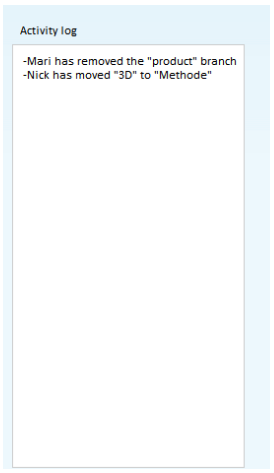
To avoid conflicts in the scenario where two users are trying to move the same concept in a taxonomy, we could consider pushing a message to the other users that this concept is being moved. The concept must be locked on the server by the first user. The second user sees a message that the concept is locked as soon as he starts dragging. If we push notifications from then onwards, other users will see when the concept is available again, and can manually refresh the tree before they start to drag the concept. But as pushing always undergoes some (network) delay, this is not failsafe.

5.3 Activity log

In this option, an activity log would be created and shown on every page, which would show which changes are being made by users in this ontology.

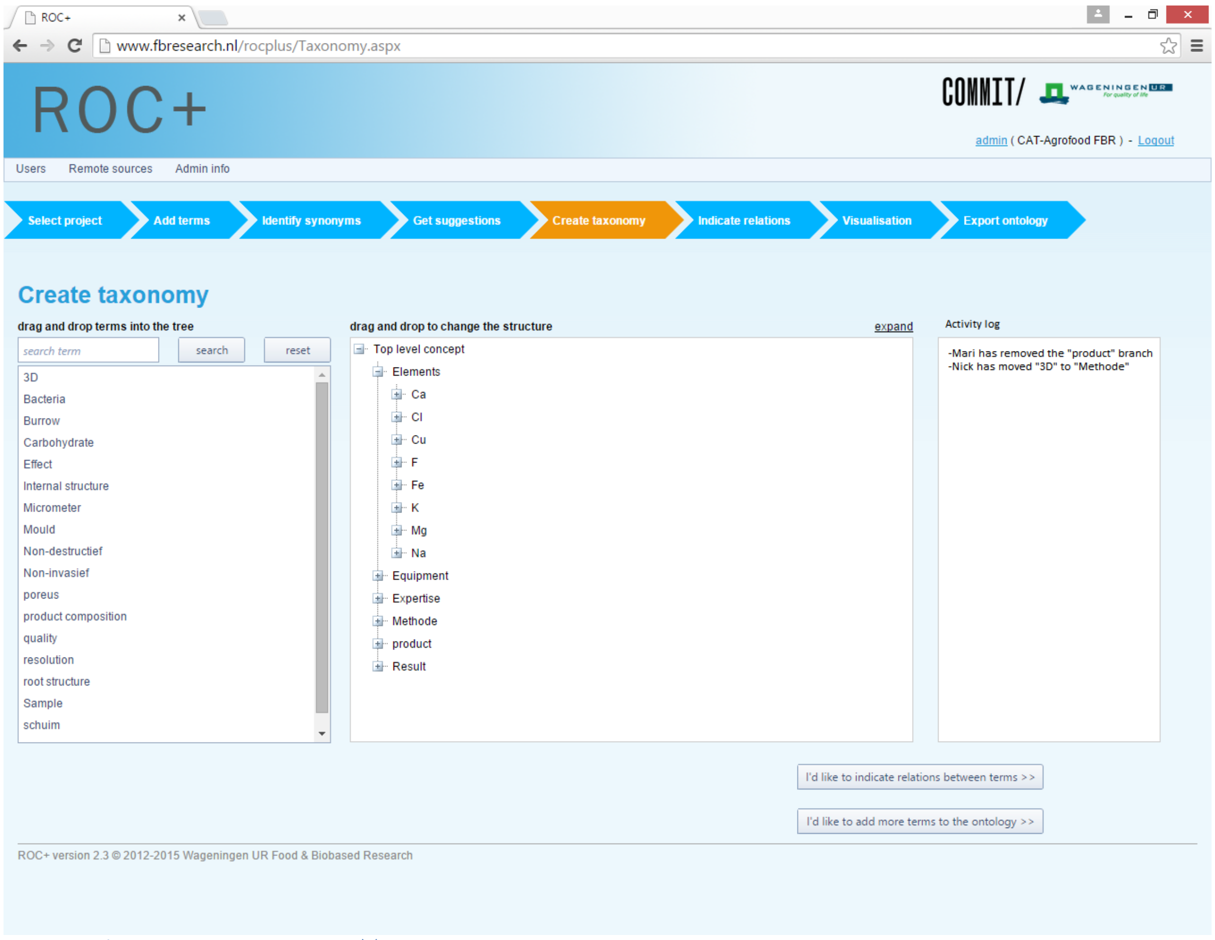
The taxonomy tree would not be refreshed in real-time, but a user can do this at any moment on their initiative, by refreshing the page. After the user does this, new information will be fetched from the server, and shown on the page.

If we consider the same example as in the real-time updates section, it would only be noticeable that someone else was dragging the concept, because a message would appear in the activity log on the screen. The taxonomy tree itself would remain unchanged, so the user can link his concept to the target concept undisturbed. As a result of the message in the log he realises that the structure has been changed, so he refreshes the taxonomy.


Activity log

-Mari has removed the "product" branch
-Nick has moved "3D" to "Methode"

Figure 1: Activity log


ROC+
www.fbresearch.nl/rocplus/Taxonomy.aspx
ROC+
COMMIT/
WAGENINGEN UR
admin (CAT-Agrofood FBR) - Logout
Users
Remote sources
Admin info
Select project
Add terms
Identify synonyms
Get suggestions
Create taxonomy
Indicate relations
Visualisation
Export ontology
drag and drop terms into the tree
search term
search
reset
drag and drop to change the structure
expand
I'd like to indicate relations between terms >>
I'd like to add more terms to the ontology >>

Create taxonomy

3D
Bacteria
Burrow
Carbohydrate
Effect
Internal structure
Micrometer
Mould
Non-destructief
Non-invasief
poreus
product composition
quality
resolution
root structure
Sample
schuim

- Top level concept
  - Elements
    - Ca
    - Cl
    - Cu
    - F
    - Fe
    - K
    - Mg
    - Na
  - Equipment
  - Expertise
  - Methode
  - product
  - Result

Activity log

-Mari has removed the "product" branch
-Nick has moved "3D" to "Methode"

ROC+ version 2.3 © 2012-2015 Wageningen UR Food & Biobased Research

Figure 2: Activity log in situ on the Taxonomy page

6 Testing

It is essential to ensure that the new, collaborative, version of ROC+ allows users to edit ontologies without any loss or corruption of data. In addition to this, the application must be easy and intuitive for experts to use. To ensure this, we will extensively test the new version of ROC+. In addition to manual testing, we will write automatic tests to test concurrency. This can be done by creating tests in multiple threads, which carry out conflicting actions. This allows a thorough and repeatable test schedule.

7 Conclusion

Implementing consistency checking and the Unit of Work concept should allow multiple users to use ROC+ at the same time, without data corruption. Error handling will ensure that users understand when something goes wrong.

For notifying users about the actions of others, we will proceed with a combination of the real-time and activity log approaches. For aspects which do not cause a large change on the screen, such as the editing of the name of a concept, we will opt for real-time, as far as possible, so that users are kept up-to-date. For aspects which do cause a large change on the screen, such as drag-and-drop, we will provide information via the activity log, so that users are aware of what is going on, without having their screen cluttered with the movements of every other user.

The next step is to investigate the best technological implementation for this design. Factors such as performance, and the extent to which the existing application will need to be rebuilt, will play a role in choosing the implementation. This implementation will then be thoroughly tested, and evaluated by users.

This design does not address more advanced aspects of collaboration, such a version control or histories. These may be tackled in the future but, at present, the focus is on enabling error-free, user-friendly collaborative editing of ontologies.