

Generalizing B-Fabric towards an Infrastructure for Collaborative Research in Switzerland

Deliverable No D5:

Specification of the Generalized Annotation Management

December 2010

Document Information

Project	
Project Title	Generalizing B-Fabric towards an Infrastructure for Collaborative Research in Switzerland
Project Start	01.06.2009
Project Sponsor	SWITCH (AAA/SWITCH Project)
Project Number	UZH.5

Document	
Title	Specification of the Generalized Annotation Management
Date	31.10.2010
Author(s)	Fuat Akal, Can Türker
Total Number of Pages	14
File Name	D5.docx
Key words	annotation management

Table of Contents

1	Introduction	5
2	Introducing Annotation Managers.....	6
3	Importing Annotations	7
3.1	Import Annotations from a Local File.....	7
3.2	Import Annotations from External Ontologies.....	8
3.2.1	Import Annotations Directly into B-Fabric vocabulary.....	8
3.2.2	Import Annotations Indirectly into B-Fabric vocabulary.....	9
4	Search Providers.....	10
5	Handling Complex Annotated Objects.....	11
6	Conclusion.....	14

Summary

In B-Fabric, users annotate their experimental data by using attribute vocabularies. These vocabularies can be extended by the users. Newly added annotations are then reviewed and released by FGCZ experts. We foresee two functionalities that should be provided by B-Fabric while extending the vocabularies. First, any eligible user other than FGCZ experts should be able to be defined as an annotation manager so that the review and release process can be done in a broader application of B-Fabric. Second, users must be allowed to select annotations from standard vocabulary repositories and to import them into B-Fabric.

In B-Fabric, experimental data is annotated by users by using the concept of samples and extracts. Samples and extracts consist of a number of attributes each of which has its own vocabulary. There are hierarchical relationships between samples and extracts, which result in complex annotated objects. Samples and extracts are organized under projects. In order to facilitate reuse of annotated sample and extract data, we define copy and move operations for samples and extracts. This includes copying and moving of hierarchies within or cross projects.

This document sketches the *i*) assigning annotation manager role to desired users, *ii*) importing external annotations into B-Fabric and *iii*) copying and moving of sample/extract hierarchies.

1 Introduction

In B-Fabric, experimental data is annotated by creating sample and extract objects. A sample contains general information about the biological source while an extract represents an extraction of that source which actually is used for the experiment or measurement. Each sample and extracts consist of a number of attributes (annotations) each of which has its own vocabulary.

In its current implementation, B-Fabric permits that a user can extend the vocabulary of certain attribute domains at run-time. That is, a user can add a new term to the vocabulary of any attribute. Once a new attribute is added to a vocabulary, an employee of the FGCZ, is triggered to review and release the new annotation in the background.

The current implementation of the vocabulary extension procedure lacks of two important functionalities. First, for a broader application of B-Fabric in a setting where manual reviewing of the vocabulary cannot be done by a small set of users, it is required to generalize the current concept such that reviewing and releasing can be done by any user who gets the right for reviewing. The granting and revoking of such rights should be possible at run-time. Second, synchronization of the annotations with standard vocabularies must be allowed. This should be done whenever possible. To do so, extensions are needed to import annotations from public standard vocabulary repositories (e.g. Gene Ontology GO) into B-Fabric.

Samples and extracts represent a hierarchical relationship. There might be several extracts of one sample. These extracts might be the result of different extraction procedures. Using these hierarchical relationships complex annotated objects can evolve. To ease the finding and reuse of sample and extract information, each sample and indirectly each extract are associated with a project. Users often want to reuse and adapt existing sample and extract information in other projects. Currently, this is not supported by B-Fabric. In addition to copy and move single samples and extracts, users demand for copying and moving entire sample and extract hierarchies. Therefore, such a copy as well as a move functionality must be available in B-Fabric.

2 Introducing Annotation Managers

B-Fabric employs a role-based access model (see the Deliverable D3: *Specification of a role-based access model for B-Fabric* for details). Roles determine what the users are authorized to do. That is, users are allowed to read or write data, provided that they have the required role.

In order to define a user as an annotation manager, we introduce a new role in B-Fabric called *annotationManager*. This role can be granted to a B-Fabric user by using the Edit User screen of B-Fabric as in Figure 1.

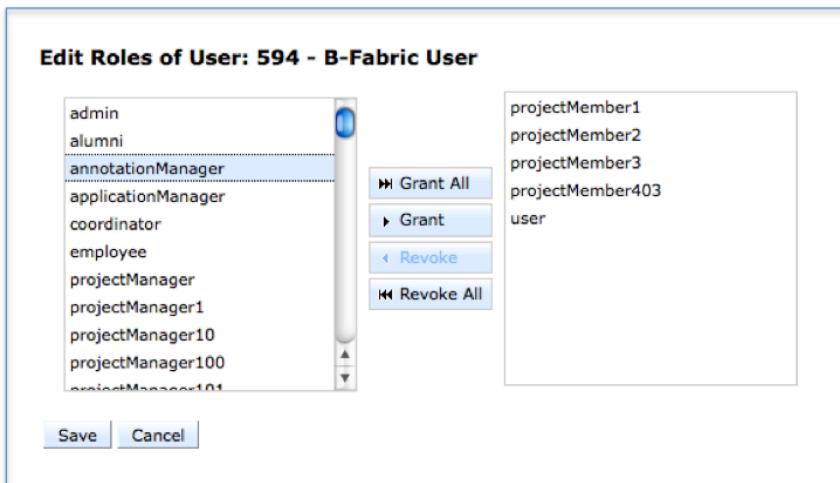


Figure 1 Managing Roles for a B-Fabric User

Once the user is granted with the *annotationManager* role, he can review and release the newly added annotations into B-Fabric. This task is achieved through Edit Annotation screen of B-Fabric shown in Figure 2. The reviewer typically checks if the annotation is spelled correctly and proper for the selected type. He also checks the description of the annotation and revises it if necessary. When all checks are complete, the annotation is released for use by saving the annotation after having the released checkbox field ticked.

In order this task to be done by a user with the *annotationManager* role, authorization before accessing the Edit Annotation screen must be adapted in B-Fabric. Furthermore, the B-Fabric code must be adapted in a way that all annotation-related functions and screens are explicitly checked via the *annotationManager* role and not via the *employee* role (refers to FGCZ experts) which was sufficient in the past.

The screenshot shows a 'Edit Annotation' dialog box. The 'Type' dropdown is set to 'Treatment'. The 'Name' field contains 'light treatment'. Under the 'Available' section, 'Not Applicable' has an unchecked checkbox. In the 'Common' section, both 'Top' and 'Common' checkboxes are checked. In the 'Released' section, both 'Released' and 'Approved' checkboxes are checked. A large text area labeled 'Description' contains the text: 'Illumination with light normally can induce a specific cell program.' At the bottom left are 'Save' and 'Cancel' buttons.

Figure 2 Editing an Annotation

3 Importing Annotations

In principle, there are two approaches for importing annotations into B-Fabric. First, annotations can be imported from specially formatted local import files. This approach is convenient for batch import of several annotations at once. Second, annotations can be fetched from external repositories. This approach provides search functionality and is convenient for batch import as well.

3.1 Import Annotations from a Local File

Annotations can be imported into B-Fabric by using specially formatted import files. The import file must provide the attributes listed in Table 1 in the given order delimited by "|" character.

Type	Type of the annotation
Name	Name for the annotation term
Not Applicable / Available	This flag indicates the annotation stands for 'not available'
Top	This flag indicates whether the annotation should be listed top in the selection lists
Common	This flag describes the most common values
Released	The annotation was checked/curated by an FGCZ employee and thus released for usage to public
Description	Description of the annotation
Ontology	External ontology this annotation comes from
TermId	Id of the ontology term

Table 1 Import File Attributes

3.2 Import Annotations from External Ontologies

When importing annotations from external ontologies, we foresee two different cases. The first case is to directly import annotations from an external ontology. This operation is performed by a user with the *annotationManager* role and often used to add several annotations at once to the B-Fabric vocabulary. The second case occurs spontaneously. For instance, when a user wants to a new annotation that does not exist in B-Fabric's vocabulary while creating a new sample.

3.2.1 Import Annotations Directly into B-Fabric vocabulary

For direct importing of annotations, the following adaptations and extensions to B-Fabric have to be carried out:

- A form to search over selected ontologies and to add found terms into B-Fabric vocabulary has to be implemented (as similar to Figure 3).
- Search providers must be implemented for external ontologies. We intend to support only GO ontologies at the moment. However, the system will be modular and any other external ontology that provides proper interfaces (for search and retrieve) should be easily pluggable.

Import Annotation

Ontology: GO (Gene Ontology)

Search: cytopla%

Terms:

- GO:0005737 cytoplasm
- GO:0007028 cytoplasm organization
- GO:0000789 cytoplasmic chromatin
- GO:0000229 cytoplasmic chromosome

Select All | Select | Deselect | Deselect All

Save | Cancel

Figure 3 Direct importing of annotations

3.2.2 Import Annotations Indirectly into B-Fabric vocabulary

For indirect importing of annotations, the following adaptations and extensions to B-Fabric have to be carried out:

- Adding user defined annotations is already possible with B-Fabric. This functionality sketched in Figure 4 will be extended by introducing a new button called “Select from Ontology”.

The screenshot shows a dialog box titled "Add Annotation". It has three main sections: "Type" (set to "Cell Compartment"), "Name" (with a "Select from Ontology >>>" button), and "Description" (an empty text area). At the bottom are "Save" and "Cancel" buttons.

Figure 4 Indirect importing of annotations

- Pressing this button means that the user wants to select a single annotation from some ontology and to add it to the B-Fabric vocabulary. To achieve this, a form as seen in Figure 5 has to be implemented. This form will allow the user to search over some ontology for a specific term and, if found, to add it to the B-Fabric vocabulary.

The screenshot shows a dialog box titled "Import Annotation". It has three main sections: "Ontology" (set to "GO (Gene Ontology)" with a dropdown arrow), "Search" (containing the text "cytopla%" and a magnifying glass icon), and "Terms" (listing four GO terms: GO:0005737 cytoplasm, GO:0007028 cytoplasm organization, GO:0000789 cytoplasmic chromatin, and GO:0000229 cytoplasmic chromosome). At the bottom are "Save" and "Cancel" buttons.

Figure 5 Importing a single annotation from an external ontology

4 Search Providers

Importing an annotation from an external ontology is performed through search providers. A search provider is simply an interface to an ontology through which a search for the desired annotation is performed. Figure 6 shows the architecture for using search providers to access external ontologies. The search provider is a software component added on to B-Fabric to access a specific ontology.

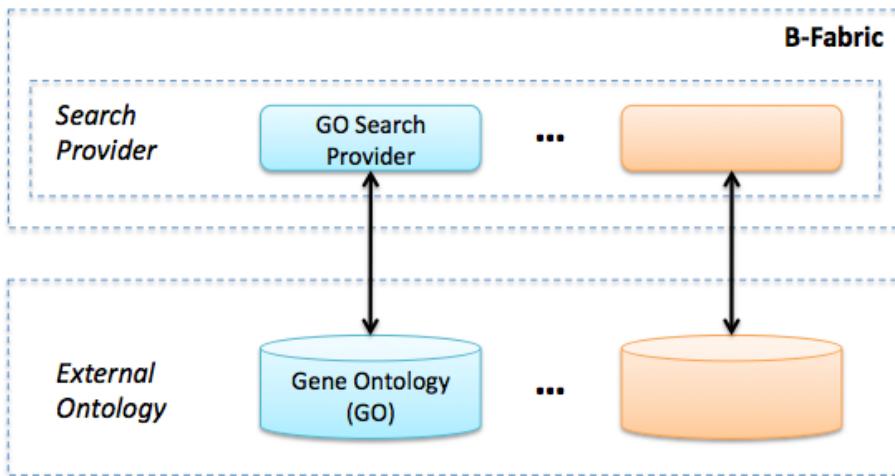


Figure 6 Search Providers

The following functions have to be implemented for every search provider:

```
public List<Object> search(String term) throws Exception  
@param term Search term which can be a regular expression  
Searches for a search term in the ontology and returns a list of object which contains found term, term id in the ontology.
```

```
public String descriptionURL(String term) throws Exception  
@param term Exact annotation name  
Searches for the annotation name in the ontology and returns the url to the description of the term.
```

```
public String descriptionURL(String termId) throws Exception
```

```
@param term Term id of the annotation in the related ontology
```

Returns the url to the description of the term if exists. No search in the ontology is necessary as the provider already knows how to generate description URLs by using term ids.

5 Handling Complex Annotated Objects

Figure 6 sketches the core UML classes implemented in B-Fabric. Samples are bound to projects, extracts as well via their parent sample. Each extract has either a parent sample or one or more parent extracts from which it is derived. The latter is used to model extracts mixed from several others extracts (imagine mixing fluids).

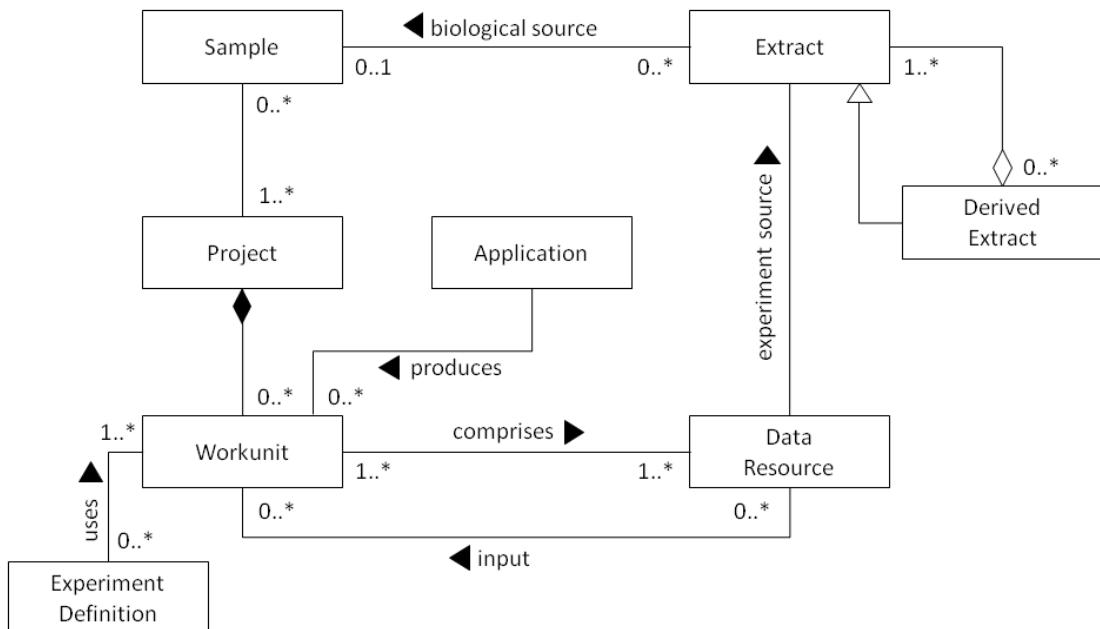


Figure 7 UML Diagram with B-Fabric Core Classes

Since extracts can be derived from one or more (parent) extracts, entire extract hierarchies can evolve (see Figure 6).

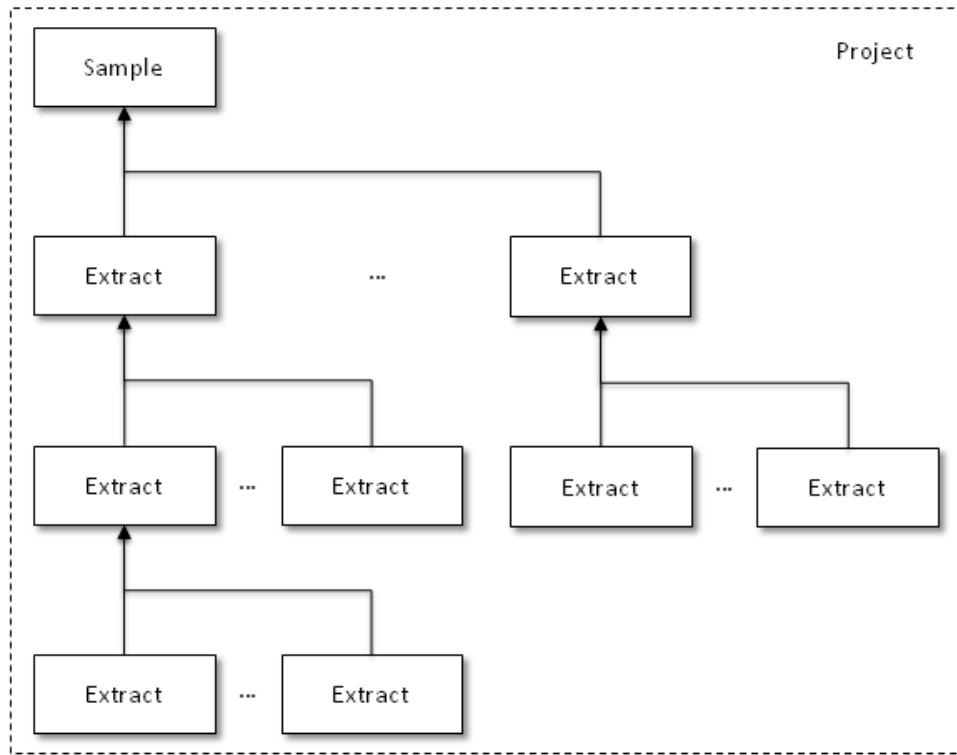


Figure 8 Sample / Extract Hierarchy

Often, users want to reuse and adapt existing sample and extract information in other projects. Currently, this is not supported by B-Fabric. In addition to copy and move single samples and extracts, users demand for copying and moving entire sample and extract hierarchies. Therefore, the plan is to implement such a copy and a move functionality in B-Fabric:

1. *Shallow Sample Copy*: Copy only a single sample object without any extracts derived from this sample.
2. *Deep Sample Copy*: Copy a single sample object together with the entire extract object hierarchy (transitively) derived from this sample object.
3. *Shallow Extract Copy*: Copy an extract object together with all its ancestors, i.e., the parent samples and parent extracts (see Figure 7).
4. *Deep Extract Copy*: Copy an extract object together with its ancestors, i.e., its parent samples and parent extracts, and all its descendants, i.e., derived child extracts (see Figure 8).
5. *Sample Move*: Move a sample to a project together with its entire extract hierarchy.

6. *Extract Move*: Move an extract together with its entire child extract hierarchy. There are two options for handling the ancestors of this sample:

- Move also all ancestors to the new project. This option is possible if there is no other extract which is derived from any of the ancestors.
- Copy all ancestors to the new project. This option is always possible.

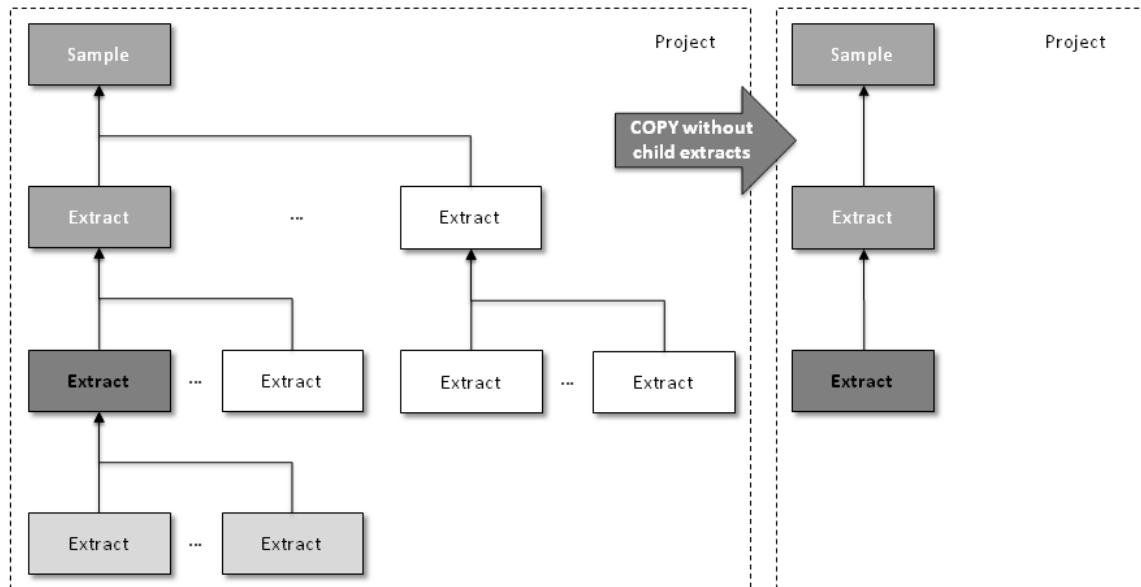


Figure 9 Copy Extract without Child Extracts

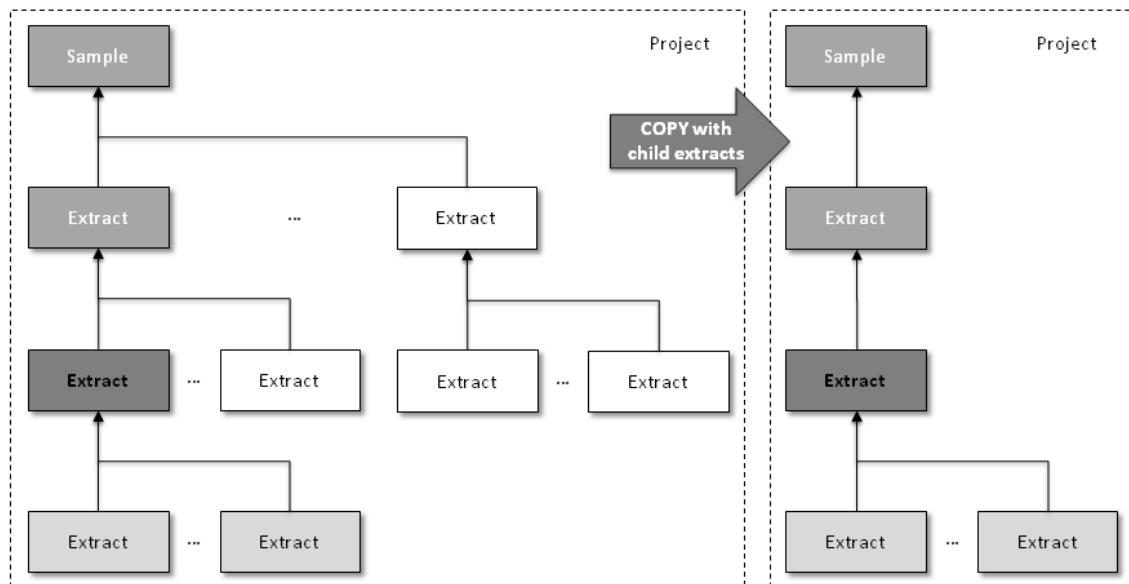


Figure 10 Copy Extract with Child Extracts

Note that extracts can be bound to resources (see again Figure 5). This is the case, after a workunit has been created and its resources have been assigned to extracts. In such cases, an extract move cannot be supported without requiring the movement of the underlying resources.

6 Conclusion

In this delivery, we have sketched three extensions of the current B-Fabric annotation framework which will be implemented within this project, namely:

1. Connecting B-Fabric annotations with terms of external ontologies
2. Importing annotation to B-Fabric via different interfaces
3. Copying and moving sample/extract hierarchies