# Seventh Framework Programme
# ICT-2009-6.4
# Information and Communication Technology


# Tagging Tool based on a Semantic Discovery Framework

**Project ID: 247893**

## Deliverable D3.1.2d

**Version 1.0**

## TaToo Semantic Tagging Tools and Services Specifications V2

**Annex of D3.1.2 Semantic Service Environment and Framework Architecture V2**

# Document Control Page

| | |
|---|---|
| **Title** | TaToo Semantic Tagging Tools and Services Specifications V2 |
| **Creator** | cismet GmbH (CIS) |
| **Editor** | Pascal Dihé |
| **Description** | This Annex to D3.1.2 contains the functional specifications of the TaToo Semantic Tagging Tools and Services. |
| **Publisher** | TaToo Consortium |
| **Contributors** | all |
| **Type** | Text |
| **Format** | MS Word |
| **Language** | EN-GB |
| **Creation date** | 09.05.2011 |
| **Version number** | 1.0 |
| **Version date** | 30.07.2011 |
| **Last modified by** | Pascal Dihé |
| **Rights** | Copyright “TaToo Consortium”.<br>During the drafting process, access is generally limited to the TaToo Partners. |
| **Audience** | ☐ internal<br>☒ public<br>☐ restricted,<br>access granted to: |
| **Review status** | ☐ Draft<br>☐ WP Leader accepted<br>☐ PCO quality controlled<br>☒ Co-ordinator accepted<br>Where applicable:<br>☐ Accepted by the GA<br>☐ Accepted by the GA as public document |
| **Action requested** | ☐ to be revised by Partners involved in the preparation of the Project Deliverable<br>☐ to be revised by all TaToo Partners<br>☐ for approval of the WP Leader<br>☐ for approval of the PCO (Quality Manager)<br>☐ for approval of the Project Co-ordinator<br>☐ for approval of the General Assembly |
| **Requested deadline** | - |

| Revision history | | | |
|---|---|---|---|
| **Version** | **Date** | **Modified by** | **Comments** |
| 0.1 | 09/05/2011 | PDi | 1st draft version, copied specifications from D3.2.1 |
| 0.2 | 26/05/2011 | PDi | + Schema Mapping & RDF Tagger |
| 0.3 | 27.07.2011 | SNe | + Evaluation Service + Processor |
| 0.4 | 30.07.2011 | PDi | Tagging Updates |
| 0.8 | 31.07.2011 | PDi | Finalised for QA |
| 0.9 | 08.08.2011 | GSc, TPa | QA |
| 1.0 | 09.08.2011 | PDi | Corrections after QA |
| 1.0 | 22.08.2011 | GSc | Accepted by PMO |
| | | | |

 

---

# Table of Contents



 

## Index of Figures



## Index of Tables

# 1. Management summary

The present document has been produced by the consortium of the European Project FP7-247893 Tagging Tool based on a Semantic Discovery Framework (TaToo) and corresponds to Annex 2 of the deliverable D3.1.2 – Semantic Service Environment and Framework Architecture V2 (TaToo-D312, 2011).

In this annex the focus is on the tagging and evaluation related components, which are described in detail together with a functional specification intended to be used by software developers. The deliverable, as the second version of three iteration cycles, provides the functional specifications of the:

- Tagging Portlet
- Tagging Service
- RDF Tagger Component
- Schema Mapping Component
- Visualisation & Filtering Component
- Evaluation Service
- Evaluation Processor

Presented components are specified using common specification templates worked out in the context of WP3 - Specification, see TaToo-D312a, 2011 and TaToo-D312b, 2011.

The achievements of the second version and the main improvements compared to the first version of the Semantic Tagging Tools and Services Specifications are:

- Specification of the Evaluation Service and the Evaluation Processor.
- Specification of the RDF Tagger Component
- Update of the Specification Tagging Service and Schema Mapping Component.
- Integration of multilingualism into the discovery lifecycle.

Possible topics of interest for the next version that have been identified are:

- Specification of additional Tagging Portlets, e.g. a portlet for geo-tagging.

## 1.1. Purpose of this document

The goal of this document is to provide the updated and new functional (implementation independent) specifications of TaToo tagging and validation components and services according to the guidelines provided in D3.1.2 – TaToo Semantic Service Environment and Framework Architecture (TaToo - D312, 2012) while considering the revised functional and non-functional requirements identified in D2.3.2 – Requirements Document (TaToo - D232, 2011) and the experience from the first implementation and design phase which took place in WP4 - Implementation.

## 1.2. Intended audience

The target readers of this document are individuals interested in the TaToo Project, especially in the tagging and validation functionality, as well as Work Package and Task Leaders of the TaToo project (WP4 - Implementation, WP5 – Validation Scenarios) involved in the implementation of TaToo Services and Tools related with tagging.

## 1.3. Structure of the document

In the following an overview of the document structure and the relationships between the different chapters is given.

- **Chapter 1** consists of this executive summary, it offers and overview and explains the overall purpose of this document.
- **Chapter 2** provides definitions for technology standards referred in subsequent chapters.
- **Chapter 3** contains the updated specifications of tagging user components for the second iteration according the guidelines provided in D3.1.2 – TaToo Semantic Service Environment and Framework Architecture (TaToo - D312, 2011).
- **Chapter 4** contains the specification of tagging public services for the second iteration according the guidelines provided in D3.1.2 – TaToo Semantic Service Environment and Framework Architecture (TaToo - D312, 2011).
- **Chapter 5** contains the specification of tagging processors (core components) for the second iteration according the guidelines provided in D3.1.2 – TaToo Semantic Service Environment and Framework Architecture (TaToo - D312, 2011).
- **Chapter 6** summarizes the results of the work performed so far in the scope of tagging.
- **Chapter 7** recognizes that research was funded by the European Community.
- **Chapter 8** lists the references and bibliography used in writing this document.

 

# 2. Referenced standards

With the aim of improving document readability, the description of the standards referred across sections describing components have been compiled here.

**Java Portlet Specification (JSR 168 and 268)**

Portlets are defined as Java-based Web components, managed by a portlet container, which processes requests and generates dynamic content. Portals use portlets as pluggable user interface components that provide a presentation layer to information systems. The Java Portlet Specification achieves interoperability among portlets and portals by defining the APIs for portlets and by standardizing the rules for preferences, user data, portlet requests and responses, deployment, packaging, and security.

**OWL**

OWL is a W3C recommendation for a language for defining ontologies performed as a vocabulary extension of RDF. The specification of OWL defines two subsets identified as relevant to the implementers. These subsets are OWL-Lite (for a basic implementation) and OWL-DL (providing the same expressiveness of description logics). However, there are also other well-known subsets defined by different implementers of OWL.

**RDF**

RDF is a W3C recommendation to establish a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed.

**WSDL**

The Web Service Description Language, in version 2.0, is a W3C recommendation for describing Web services. According to the W3C definition, WSDL provides a model and an XML format for describing Web services. WSDL enables one to separate the description of the abstract functionality offered by a service from concrete details of a service description such as “how” and “where” that functionality is offered. This specification defines a language for describing the abstract functionality of a service as well as a framework for describing the concrete details of a service description. It also defines the conformance criteria for documents in this language. Finally, WSDL also describes extensions for message exchange patterns, operation safety, operation styles and binding extensions for SOAP and HTTP.

**REST**

Although not yet a W3C Recommendation, REST based Web services have had a good adoption since its inception in 2000, being included in the agenda of the W3C working group dedicated to Web services. The Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

 

The key concept in REST is the existence of resources (sources of specific information), each of which is referenced with a global identifier (e.g., a URI in HTTP). In order to manipulate these resources, components of the network (user agents and origin servers) communicate via a standardized interface (e.g., HTTP) and exchange representations of these resources (the actual documents conveying the information). A REST based web service is a simple web service implemented using HTTP and the principles of REST. It is a collection of resources, with three defined aspects: the base URI for the web service, the MIME type of the data supported by the web service and the set of operations supported by the web service using HTTP methods (e.g., POST, GET, PUT or DELETE).

**SPARQL**

SPARQL is the query language for RDF repositories recommended by the W3C. SPARQL allows users to query on different RDF repositories using a common language. SPARQL queries describe patterns of graphs that are matched against a repository. The result of a SPARQL query can be a result set or a RDF graph. SPARQL allows performing set operations over its results (union, intersection, etc).

**XSLT**

XSLT is a language for transforming XML documents into other XML documents by the W3C. It "is a declarative, XML-based language used for the transformation of XML documents into other XML documents. The original document is not changed; rather, a new document is created based on the content of an existing one." (http://en.wikipedia.org/wiki/XSLT).

 

# 3. User Components

This chapter describes in details the Tagging and Evaluation tools.

## 3.1. Tagging portlet

This section presents a detailed description and specification for the Tagging Portlet. It provides a component overview followed by the description of the relationships between the component and its context.

### 3.1.1 Overview and outline

This section provides an overview of the component describing the nature and the role of the component.

#### 3.1.1.1 Nature of the component

The Tagging Portlet is a TaToo User Component. It is part of the TaToo Web Portal and provides tagging functionality to the end user (in particular to taggers), with the Tagging Services to access the TaToo Business tier. Portlets are defined as Web components, managed by a portlet container, that process requests and generate dynamic content. Portals are configurable through different portlets to offer the presentation layer to the end user.

The concrete realisation of the Tagging Portlet may depend on the meta-information schema (tags) to be supported as well as on specific user requirements. As a starting point the Tagging Portlet will be realised as a generic tagging tool providing basic user interface and functionality.

#### 3.1.1.2 Role and scope of the component

The Tagging Portlet allows a user to tag information about a discovered or already known resource, where a resource could be a data source, a service, a Web page, etc. The user is prompted by the portlet with a selection panel to choose terms from an ontology to create tags for a resource or a set of resources. When the user adds tags, the Tagging Portlet contacts the TaToo Tagging Service to update the information related to the resource in the TaToo Framework Repository (e.g. meta-information repository).

The Tagging Portlet displays to the TaToo Web Portal user the result of the tagging operation whether it is successful or unsuccessful.

 

### 3.1.2 Context

This section describes the relationships between the component and its context, including technical requirements, other TaToo Components, etc.

#### 3.1.2.1 Relation to technical requirements

The Tagging Portlet component addresses the following technical requirements as specified in D2.3.2 – Requirements Document V2 (TaToo-D232). The technical requirements are mapped to the concrete functional requirements specified in chapter 3.1.3.3.

| Requirements ID and Name | Scope | Functional Requirement | Comments |
|---|---|---|---|
| TR.TAGGING.010 Meta-information on third party resources | S/I | 2, 3, 4 | |
| TR.TAGGING.020 Access to tags (TaToos) | S/I | 5 | |
| TR.TAGGING.030 Postponed Tagging / Tagging of known resources | S/I | n/a | Requirement on the Tagging Service.. |
| TR.TAGGING.050 Tagging Client | S/I | all | The portlet itself is a client. |
| TR.TAGGING.090 Sharing of Tags (TaToos) | S/I | 5 | |
| TR.TAGGING.100 Editing of Tags (TaToos) | S/I | n/a | Handled by dedicated tags editing portlet |

**Table 3-1: Tagging Portlet technical requirements**

#### 3.1.2.2 Relations to standards

The Tagging Portlet must be specified and implemented following the Java Portlet Specification 168 and 268. A brief description of the Java Portlet Specification can be found in chapter 2, Referenced standards.

The Tagging Portlet processes the tags (TaToos) provided by the user as selected from an Ontology. Available ontologies are encoded using OWL. A brief description of OWL can be found in chapter 2, Referenced standards.

The Tagging Portlet returns the tags selected from the user to the Tagging Services in XML format. A brief description of XML can be found in chapter 2, Referenced standards.

#### 3.1.2.3 Relations to other TaToo components

The TaToo Tagging Portlet depends on the TaToo Tagging Service to process the tagged information and update the meta-information related to the particular resource in the TaToo Repository.

The TaToo Tagging Portlet depends on the Search Portlet for the discovery of the resources to be tagged, if already existing.

#### 3.1.2.4 Relations to information models

The Tagging Portlet should be able to offer the tagging functionality to the user no matter what used ontology. The ontology should serve the user providing terms to tag the resources. In this process the Tagging Portlet can take advantage of the MERM.

More information about ontologies and MERM can be found in D3.1.2 – TaToo Semantic Service Environment and Framework Architecture V2, Chapter 6.4 - Minimum Environmental Resource Model.

### 3.1.3 Specification

This section provides the functional specification of the Tagging Portlet, including objectives, mock ups and requirements.

#### 3.1.3.1 Objectives

The objective of the Tagging Portlet is to give the user a graphical interface that allows to:

- add annotations, intended as a semantically enriched tags, to a resource;
- add tags to more than one resource matching a search query;
- display tags already associated to a known resource.

#### 3.1.3.2 Target users

Tagging Portlet users are those interested in adding tags to already know resources or as resulting from a discovery process. In the former case the users provide the URIs identifying the resources they already know; this is generally the case where the user is the resource owner. In the latter case the user searches resources through a query (through the Search & Discovery Portlet) and tags those resources they consider relevant.

#### 3.1.3.3 Functional requirements

The functional requirements for the Tagging Portlet are:

1. Terms Selection. Allows the user to select a term from an ontology prompted by the portlet;
2. Tagging. Adds the term to the meta-information associated to the resource;
3. Multiple Tagging. Allows to tag multiple resources at the same time with the same terms;
4. Multiple Resources Tagging. Allows the user to tag at the same time with different terms multiple resources;
5. Retrieve Tagging. Allows the user to access already known resource and update existing tags;
6. Tagging Result. Informs the user about the result of the tagging operation.

#### 3.1.3.4 Mock up

In this section a mock up of the Tagging Portlet is presented (see Figure 3-1). The Tagging Portlet is contained within the TaToo Web Portal, and accessed through it. From left to right the portlet presents three working panels that the user can access to perform the operations of tagging provided by the TaToo Framework.

In the first panel ('RESOURCES') the user has the option to add a new resource providing its URI through a text field. At the bottom part, a list of provided / discovered resources is shown. For each resource there are two drop down lists of items: 'Current Annotations', that lists the meta-information already associated with the resource, and 'New Annotations', that lists the meta-information that the user is currently adding during the portlet working session. To add a new annotation, the user clicks on the 'Add' link next to the 'New Annotations' label. This adds a new URI to be valorised by the user.

In the second panel, bottom right, the meta-information the user can display for the 'Current Annotations' and edit for the 'New Annotations' is provided. The annotation is structured in three parts: Resource URI, Property and Value, where Property and Value are specified selecting a term from an ontology (respectively predicate and object in the RDF triple).

 

In the third panel, upper right, there are two trees representing two different ontologies. The user can browse the ontologies respectively to select the term for Property and the term for Value.

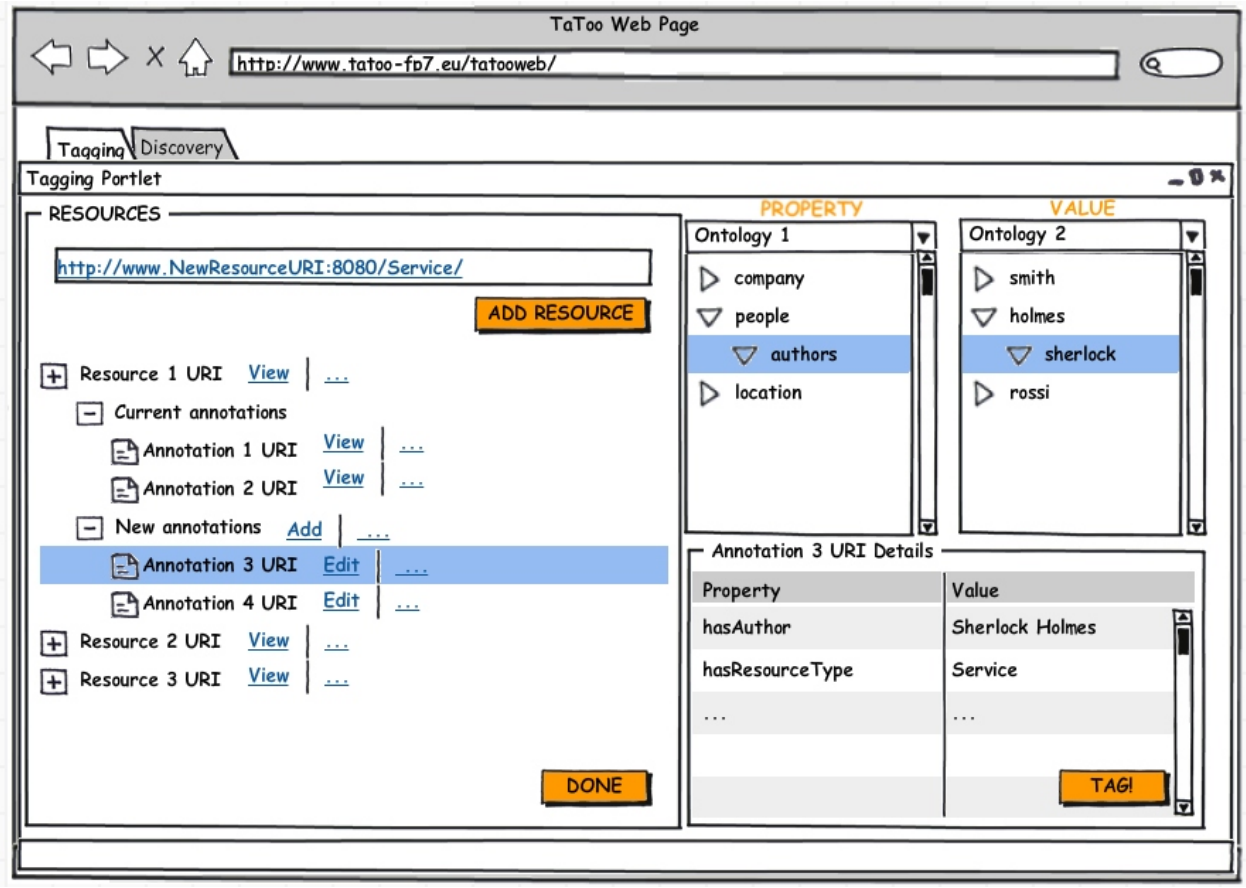


*Figure 3-1: Tagging Portlet Mockup*

Other choices are possible for instance in the case the ontologies are too complicated and the user should be supported somehow in concepts identification and selection.

Figure 3-2 shows an alternative mock up considering combo boxes to show concepts the user can select as tags (property and value). Concepts to show to the user and to make available can be identified depending on user's role through a profile. A profile can be defined per role (or user) where available concepts for the properties and possible value are listed.

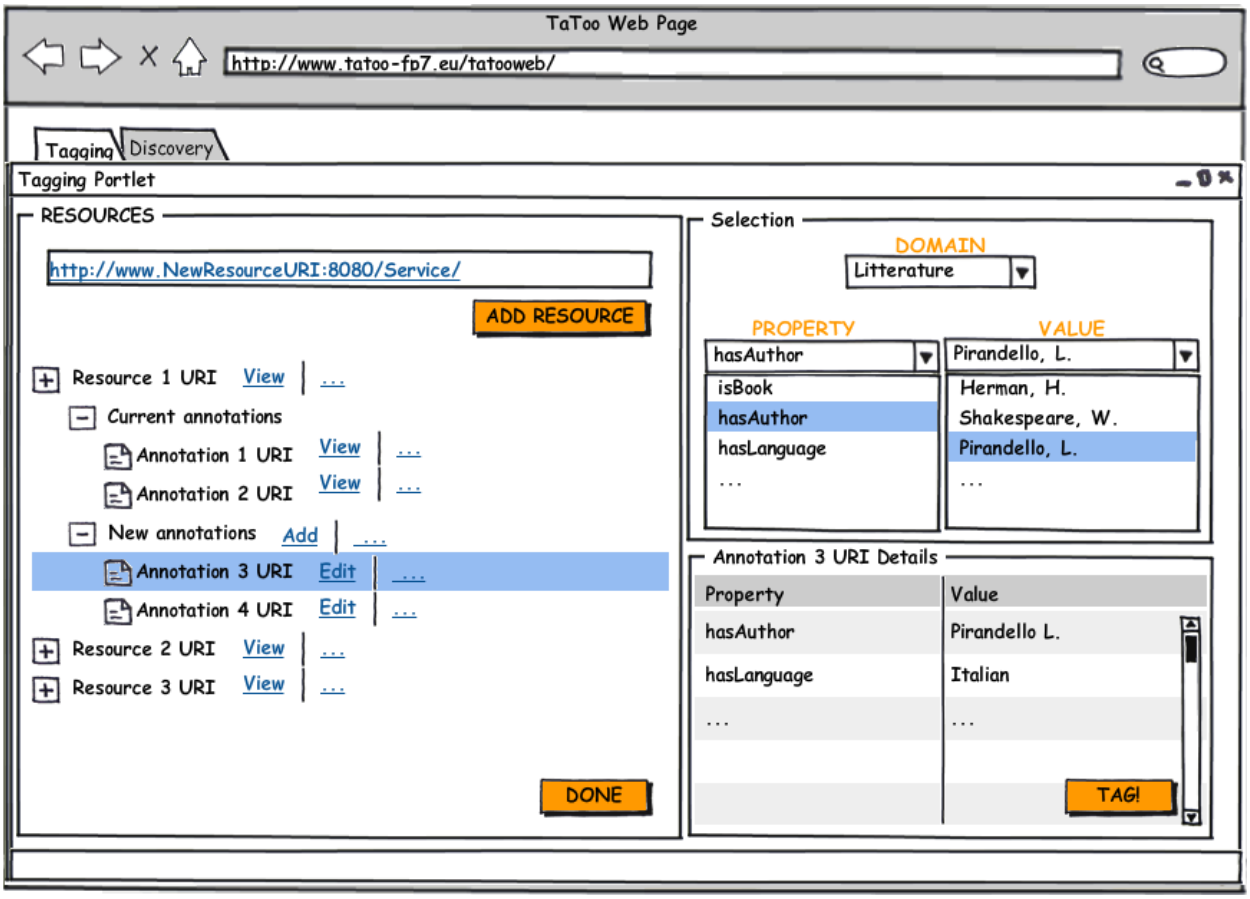


*Figure 3-2: Tagging Portlet Mockup based on Profile*

#### 3.1.3.5 Interaction with other TaToo components

The Tagging Portlet interacts with the Tagging Service to retrieve the ontology that the user wants to use to annotate the resource. It also interacts with the Search Portlet to get discovered resources as resulting from a user's search process.

The Tagging Portlet interacts with the Tagging Service to:

- annotate semantically enriched meta-information to the selected resource;
- add a common set of annotations to more than one resource;
- add various annotations to more than one resource;
- retrieve annotations of a known resource;
- retrieve the ontology of a specific domain;
- retrieve the MERM ontology.

# 4. Public Services

This chapter includes the specification of the Tagging and evaluation services.

## 4.1. Tagging Service

This section defines the functional specification of the Tagging Service and is based on the Template for the Specification of TaToo Service.

In its current version the Tagging Service supports the modification and removal of previously created tags. This functionality was added in V2.

Note: The concrete realisation of functionality depends on several implementation decisions, which haven't been taken yet. Deleting tags physically from the knowledgebase could lead to several unwanted side effects on inferred tags as well as on evaluations. There the possibility to mark a tag as "deprecated" is currently under consideration. The same applies in principle to the tags editing functionality.

### 4.1.1 Overview and Outline

This section provides an overview of the component describing the Nature and the Role of the component.

#### 4.1.1.1 Role and Scope of the Service

The Tagging Service is a TaToo Public Service that allows User Components, in particular the Tagging Portlet, to access the tagging functionality offered by the TaToo System and exposed through the public interface of this service. A Tagging Service receives tagging requests from the different User Components to:

- associate tags with resources;
- access already tags;
- update tags associated with the resource;
- delete tags;
- access ontologies.

In V1 the functionality of the Tagging Service was limited to the creation of new tags and the read-only access to existing tags. In V2 support for editing and deleting was added. Tagging is limited to semantic tagging which means choosing from a number of terms of the selected ontology and the supported formats for ontologies and tags are limited to RDF and OWL. The access to ontologies is realised by a dedicated Ontology Retrieval Interface.

 

Please note that the concept of deleting and editing tags is currently under discussion. The concrete realisation of the perceived functionality of the *deleteTags* and *editTags* in WP4 – Implementation depends on the following open issues which have yet to be resolved:

- If a user edits an annotation (set of tags) after some evaluations are already done for this specific annotation, this would actually invalidate all associated evaluations.
- If a user deletes an annotation, also all evaluations that are associated with this specific annotation would have to be deleted.
- Deleting or editing annotations and evaluations could lead to loss of inferred knowledge. In this case, the behaviour of the knowledge base is difficult to predict and unwanted side effects may occur.

Possible solutions to these issues could be to actually forbid the editing of tags and to flag a tag as deprecated instead of deleting it from the knowledge base. For this reason, the *deleteTags* and *editTags* operations are marked as optional as it is currently not decided whether and how this functionality is implemented.

The Tagging Service is the service responsible for all the operations related to the tagging of resources. It interacts with the tagging User Component that demands updating of tags for a resource and the Clearinghouse to access to the knowledgebase, but also to retrieve ontology information if needed. For the specification of the Clearinghouse, please refer to D3.1.2 - Semantic Service Environment and Framework Architecture V2 (TaToo-D312), section 7.3.1 - Clearinghouse.

The Tagging Service is supposed to be an interoperable and standalone Web service, in particular a Web Service W3C compliant taking advantage of widely adopted standards such as XML, WSDL, and SOAP.

#### 4.1.1.2 Service specification summary

The service specification of Tagging is comprised of the following interfaces:

- the Tagging Interface, that includes all operations related to the tagging functionality.
- the Ontology Retrieval Interface, that includes all operations related to the retrieval of ontologies to support tagging functionality.

Where the Tagging Interface contains the following operations:

- addTag, associates a single tag with exactly one resource.
- addTags, associates different tags with at least one resource.
- getTag retrieves a specific tag of exactly one resource.
- getTags retrieves all tags of at least one resources.
- editTag edits a specific tag of exactly one resource
- editTags edits all tags of at least one resource
- deleteTag deletes a specific tag of exactly one resource

- deleteTags deletes all tags of at least one resource

And the Ontology Retrieval Interface contains:

- getOntology, retrieves an ontology related to a certain domain.
- getMERMOntology, retrieves an ontology containing the MERM.

### 4.1.2 Context

This section describes the relationships between the service and its context, including technical requirements, other TaToo components, etc.

#### 4.1.2.1 Relation to technical requirements

The Tagging Service addresses the following technical requirements as specified in D2.3.2 – "Requirements Document V2".

| Requirements ID and Name | Scope | Fulfilment |
|---|---|---|
| TR.TAGGING.010 Meta-information on third party resources | S/I | by the *addTag* and *addTags* operation. |
| TR.TAGGING.020 Access to tags (TaToos) | S/I | by the *getTag* and *getTags* operation. |
| TR.TAGGING.030 Postponed Tagging / Tagging of known resources | S/I | by providing the Tagging Interface and implementing it as web service. |
| TR.TAGGING.040 Tagging Service | S/I | see above. |
| TR.TAGGING.060 Semantic Tags | S/I | by supporting tags that are terms from an ontology and by providing the Ontology Retrieval interface. |
| TR.TAGGING.080 Ontology supported tagging | S/I | see above. |
| TR.TAGGING.100 Editing of Tags (TaToos) | S/I | by the update and delete tag operations |

**Table 4-1: Tagging Service technical requirements**

#### 4.1.2.2 Relations to standards

The Tagging Service is a Web Service that will follow W3C recommendations. The Tagging Service could be either a RESTful Web Service, exposing a uniform set of "stateless" operations, or an arbitrary Web Service exposing a WSDL Interface that contains an arbitrary set of operations. A brief description of WSDL and REST can be found in chapter 3, Referenced standards.

The Tagging Service submits the ontologies to the Tagging Portlets. These ontologies are encoded using OWL, a brief description of OWL can be found in chapter 3.

The Tagging Service returns the meta-information associated by the user to the resource to the Clearinghouse as RDF-Triples. A brief description of RDF can be found in chapter 3.

#### 4.1.2.3 Relations to other TaToo components

The TaToo Tagging Service should be accessible via its interface from Tagging Tools, specifically the Tagging Portlet or other portlets and third parties applications.

The Tagging Service interacts with the TaToo Clearinghouse to access the TaToo Core Components to store the meta-information related to the resource and / or eventually process the meta-information via a TaToo Tagging Processor component.

#### 4.1.2.4 Relations to information models

The Tagging Service operations process information that relate to the MERM and other ontologies of a certain domain, in particular when sending an ontology to the Tagging Portlet to provide selectable terms to the end user.

More information about ontologies and MERM can be found in D3.1.2 - Semantic Service Environment and Framework Architecture V2 (TaToo-D312), chapter 6.4 - Minimum Environmental Resource Model.

### 4.1.3 Specification of the Tagging Service Interface

This section provides the functional specification of the Tagging Service, including the Service Interface and its operations.

The objective of the Tagging Service is to provide an interoperable, Web available interface as an entry point for the TaToo Tagging operations, which are described in the following table:

| Operation Name | Description |
|---|---|
| addTag | Optional convenience operation that associates exactly one tag (TaToo) with 1-n resources and sends it to the |

| | |
|---|---|
| | Clearinghouse. |
| addTags | Mandatory operation that adds 1-n tags (TaToos) to 1-m resources and sends it to the Clearinghouse. |
| getTag | Optional convenience operation that retrieves a specific tag associated to one resource from the TaToo Framework Repository. |
| getTags | Mandatory operation that retrieves 1-n tags associated to 1-m resources from the TaToo Framework Repository. |
| editTag | Optional operation that edits a specific tag of exactly one resource |
| editTags | Optional operation that edits all tags of at least one resource |
| deleteTag | Optional operation that deletes a specific tag of exactly one resource |
| deleteTags | Optional operation that deletes all tags of at least one resource |

**Table 4-2: Tagging Service Interface**

#### 4.1.3.1 Specification of the addTag operation

The optional *addTag* operation is responsible for the tagging of at least one resource, identified by a list of URIs, and thus allows the establishment of a 1-n relationship between one tag and a set of disparate resources. This operation supports the association of a single tag to a single resource or to multiple resources at once. In consequence, when tagging multiple resources, each resource shares the same tagging information. If different tags shall be associated with different resources at once, the *addTags* operation must be used instead. The *addTag* operation is an optional convenience operation whose functionality is covered completely by the *addTags* operation.

To achieve the tagging functionality the operation will be supported by an ontology provided by the Clearinghouse via the Ontology Retrieval Interface. In a Tagging User Component (e.g. the Tagging Portlet), the user will select the terms to compose the tag that will be associated to the resources and stored in the TaToo Framework Repository.

Please note that there is no distinct operation to add a single tag to a single resource. Instead this can be achieved by this operation by populating the list of resource URIs with exactly one element.

A request to perform the *addTag* operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**String addTag(List<URI>, RDF document) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Optional | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | resourceURIs | List<URI> | Mandatory | 1-n URIs of the resources to tag |
| | tag | RDF document | Mandatory | tag in RDF format |
| | Locale | String | Optional | Identifier of the user locale |
| | UserId | String | Optional | User unique identifier |
| **Returns** | **Type** | | **Description** | |
| | String | | Status indicator and unique id of the newly created tag | |
| **Throws** | **Type** | | **Cause** | |
| | InvalidParameterValue | | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. | |
| | MissingParameterValue | | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. | |
| | ResourcesNotFound | | The resources identified by the provided URIs could not be found. | |
| | TaTooInternalError | | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. | |

**Table 4-3: Specification of the addTag Operation**

#### 4.1.3.2 Specification of the addTags operation

The *addTags* operation is responsible for the tagging of more than one resource, identified by a list of URIs, where the tags may be heterogeneous. It thus allows the establishment of n-m relationship between tags and resources. This operation supports the association of multiple tags to a single resource or to multiple resources at once. If exactly the same tag shall be associated with one or more resources, the convenience operation *addTag* can be used instead.

The operation excepts as the single parameter a map with a URI of a resource as keys and a list of tags to be associated with the resource as values.

To achieve the multiple tagging functionality the operation will be supported by an ontology provided by the Clearinghouse via the Ontology Retrieval Interface. In a Tagging User Component (e.g. the Tagging Portlet), the user will select the terms to compose the tags that will be associated to the resources and stored in the TaToo Framework Repository.

A request to perform the *addTags* operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**String addTags(Map<URI, List<RDF Document>>) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Mandatory | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | resourceMap | Map<URI, List<RDF Document>> | Mandatory | Map with URI as keys and list of tags as values. |
| | Locale | String | Optional | Identifier of the user locale |
| | UserId | String | Optional | User unique identifier |
| **Returns** | **Type** | | **Description** | |
| | String | | Status indicator and unique id of the newly created tags. | |

| Throws | Type | Cause |
|---|---|---|
| | InvalidParameterValue | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. |
| | MissingParameterValue | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. |
| | ResourcesNotFound | The resources identified by the provided URIs could not be found. |
| | TaTooInternalError | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. |

**Table 4-4: Specification of the addTags Operation**

#### 4.1.3.3 Specification of the getTag operation

The optional *getTag* operation is responsible for the retrieval of a specific tag associated to a resource given its URI as input. This operation supports the retrieval of a single tag of a single resource. If the resource is associated to multiple tags the operation returns either the newest tag or the first tag that matches an optional filter condition. If all tags of a resource need to be retrieved at once, the *getTags* operation shall be used instead. The *getTag* operation is an optional convenience operation whose functionality is covered completely by the *getTags* operation.

The Tagging Service will contact the Clearinghouse with the URI of the resource and will receive all the meta-information. The optional filter condition is evaluated by a dedicated Tagging Processor.

A request to perform the *getTag* operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**RDF-Document getTag(Map<URI, String) throws InvalidParameterValue, TagsNotFound, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Optional | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | resourceURI | URI | Mandatory | URI of the resource. |
| | filterCondition | String | Optional | Filter for specific tags. |
| | Locale | String | Optional | Identifier of the user locale |
| | UserId | String | Optional | User unique identifier |
| **Returns** | **Type** | | **Description** | |
| | RDF-Document | | Meta-information already associated to the resource or null if the optional filter condition is not met by any tag. | |
| **Throws** | **Type** | | **Cause** | |
| | ResourcesNotFound | | The resources identified by the provided URIs could not be found. | |
| | TaTooInternalError | | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. | |
| | InvalidParameterValue | | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. | |
| | TagsNotFound | | The resource was found, but no meta-information is associated with the resource. | |

| | | |
|---|---|---|
| | MissingParameterValue | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. |

**Table 4-5: Specification of the getTag Operation**

#### 4.1.3.4 Specification of the getTags operation

The mandatory *getTags* operation is responsible for the retrieval of tags associated to resources given its URIs as input. This operation supports the retrieval of a multiple tags of multiple resources. An optional filter condition can be specified to narrow the number of tags. If only a specific tag of a single resource need to be retrieved, the convenience operation *getTag* can be used instead.

The operation returns a map with a URI of a resource as key and a list of tags associated with the resource and that match the optional filter condition as value.

The Tagging Service will contact the Clearinghouse with the URIs of the resource and will receive all the meta-information. The optional filter condition is evaluated by a dedicated Tagging Processor.

A request to perform the *getTags* operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**Map<URI, List<RDF Document>> getTags(List <URI>, String) throws InvalidParameterValue, TagsNotFound, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Mandatory | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | resourceURIs | List <URI> | Mandatory | URIs of the resource. |
| | filterCondition | String | Optional | Filter for specific tags. |

| | | | | |
|---|---|---|---|---|
| | Locale | String | Optional | Identifier of the user locale |
| | UserId | String | Optional | User unique identifier |
| **Returns** | **Type** | | **Description** | |
| | Map<URI, List<RDF Document>> | | Map with URI as keys and list of tags as values. A list of tags (value) might be empty if the optional filter condition is not met by any tag associated with the specific resource (key). | |
| **Throws** | **Type** | | **Cause** | |
| | ResourcesNotFound | | The resources identified by the provided URIs could not be found. | |
| | TaTooInternalError | | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. | |
| | InvalidParameterValue | | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. | |
| | TagsNotFound | | The resource was found, but no meta-information is associated with the resource. | |
| | MissingParameterValue | | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. | |

**Table 4-6: Specification of the getTags Operation**

#### 4.1.3.5 Specification of the deleteTag operation

The optional *deletTag* operation is responsible for the deletion of a single tag of a resource, identified by the tag URI.

A request to perform the *deleteTags* operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**String deleteTag(URI) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Optional | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | tagURI | URI | Mandatory | URI of the tag |
| | Locale | String | Optional | Identifier of the user locale |
| | UserId | String | Optional | User unique identifier |
| **Returns** | **Type** | | **Description** | |
| | String | | Status indicator and unique id of the deleted tag. | |
| **Throws** | **Type** | | **Cause** | |
| | InvalidParameterValue | | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. | |
| | MissingParameterValue | | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. | |
| | ResourcesNotFound | | The resources identified by the provided URIs could not be found. | |

| | | |
|---|---|---|
| | TaTooInternalError | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. |

**Table 4-7: Specification of the deleteTag Operation**

#### 4.1.3.6 Specification of the deleteTags operation

The optional *deleteTags* operation is responsible for the deletion of more than one tag of a resource, identified by a list of tag URIs. This operation supports the deletion of multiple tags of a single resource. If exactly one tag shall be deleted, the convenience operation *deleteTag* can be used instead. Please see also section 4.1.1.1 for possible implications caused by editing and deleting tags.

A request to perform the *deleteTags* operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**String deleteTags(List<URI>) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Optional | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | tagsList | List<URI> | Mandatory | List with tag URIs. |
| | Locale | String | Optional | Identifier of the user locale |
| | UserId | String | Optional | User unique identifier |
| **Returns** | **Type** | | **Description** | |
| | String | | Status indicator and unique id of the deleted tags. | |

| Throws | Type | Cause |
|---|---|---|
| | InvalidParameterValue | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. |
| | MissingParameterValue | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. |
| | ResourcesNotFound | The resources identified by the provided URIs could not be found. |
| | TaTooInternalError | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. |

**Table 4-8: Specification of the deleteTags Operation**

#### 4.1.3.7 Specification of the editTag operation

The optional *editTag* operation is responsible for the editing of a specific tag associated to a resource given its URI as input. This operation supports the editing of a single tag of a single resource. Please see also section 4.1.1.1 for possible implications caused by editing and deleting tags.

A request to perform the *getTag* operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**String editTag(URI, RDF document) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | |
|---|---|
| **Overrides** | Not applicable |
| **Preconditions** | None |

| | | | | |
|---|---|---|---|---|
| **Post conditions** | None | | | |
| **Use** | Optional | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | tagURI | URI | Mandatory | URI of the tag to edit |
| | tag | RDF document | Mandatory | New tag in RDF format |
| | Locale | String | Optional | Identifier of the user locale |
| | UserId | String | Optional | User unique identifier |
| **Returns** | **Type** | | **Description** | |
| | String | | Status indicator and unique id of the edited tag | |
| **Throws** | **Type** | | **Cause** | |
| | InvalidParameterValue | | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. | |
| | MissingParameterValue | | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. | |
| | ResourcesNotFound | | The resources identified by the provided URIs could not be found. | |
| | TaTooInternalError | | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. | |

**Table 4-9: Specification of the editTag Operation**

#### 4.1.3.8 Specification of the editTags operation

The optional *editTags* operation is responsible for the editing of more than one tag of a resource, identified by a list of tag URIs, where the tags may be heterogeneous. This operation supports the editing of multiple tags of a single resource. If exactly one tag shall be edited, the convenience operation *editTag* can be used instead. Please see also section 4.1.1.1 for possible implications caused by editing and deleting tags.

A request to perform the *editTags* operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**String editTags(Map<URI, List<RDF Document>>) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

<table>
<tr><td>Overrides</td><td colspan="4">Not applicable</td></tr>
<tr><td>Preconditions</td><td colspan="4">None</td></tr>
<tr><td>Post conditions</td><td colspan="4">None</td></tr>
<tr><td>Use</td><td colspan="4">Optional</td></tr>
<tr><td rowspan="4">Receives</td><td>Name</td><td>Type</td><td>Use</td><td>Description</td></tr>
<tr><td>tagsMap</td><td>Map<URI, List<RDF Document>></td><td>Mandatory</td><td>Map with tag URI as keys and list of new tags as values.</td></tr>
<tr><td>Locale</td><td>String</td><td>Optional</td><td>Identifier of the user locale</td></tr>
<tr><td>UserId</td><td>String</td><td>Optional</td><td>User unique identifier</td></tr>
<tr><td rowspan="2">Returns</td><td colspan="2">Type</td><td colspan="2">Description</td></tr>
<tr><td colspan="2">String</td><td colspan="2">Status indicator and unique id of the updated tags.</td></tr>
<tr><td rowspan="4">Throws</td><td colspan="2">Type</td><td colspan="2">Cause</td></tr>
<tr><td colspan="2">InvalidParameterValue</td><td colspan="2">Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value.</td></tr>
<tr><td colspan="2">MissingParameterValue</td><td colspan="2">Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter.</td></tr>
<tr><td colspan="2">ResourcesNotFound</td><td colspan="2">The resources identified by the provided URIs could not be found.</td></tr>
</table>

| | | |
|---|---|---|
| | TaTooInternalError | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. |

**Table 4-10: Specification of the editTags Operation**

### 4.1.4 Specification of the Ontology Retrieval interface

The ontology retrieval interface includes all operations related to ontology retrieval necessary to support search user components in TaToo. The Ontology Retrieval Interface of the Discovery Service defines the following operations:

| Operation Name | Description |
|---|---|
| getOntology | Retrieves ontologies related to a given domain |
| getMERMOntology | Retrieves the ontology containing the MERM |
| listDomains | Retrieves a list of domains loaded in the knowledge base |

**Table 4-11: Ontology Retrieval Interface operations summary**

#### 4.1.4.1 Specification of the getOntology operation

The mandatory getOntology operation is responsible of allowing external agents to retrieve ontologies used by the TaToo System. To do so, the operation will take as an input a domain identifier. The result of the operation will be the ontologies related to the given domain available in the system.

A request to perform the getOntology operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**List<OWL Document> getOntology(URI) throws InvalidParameterValue, MissingParameterValue, TaTooInternalError, OntologiesNotFound, DomainNotFound**

| | |
|---|---|
| **Overrides** | Not applicable |

| | | | | |
|---|---|---|---|---|
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Mandatory | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | Domain identifier | URI | Mandatory | An unique identifier for a domain |
| **Returns** | **Type** | | **Description** | |
| | List<OWL Document> | | All ontologies available in the system related to the given domain | |
| **Throws** | **Type** | | **Cause** | |
| | OntologiesNotFound | | There are no ontologies in the system related to the given domain | |
| | DomainNotFound | | The given domain is not registered on the system. | |
| | InvalidParameterValue | | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. | |
| | MissingParameterValue | | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. | |
| | TaTooInternalError | | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. | |

**Table 4-12: Specification of the getOntology operation**

#### 4.1.4.2 Specification of the getMERMOntology operation

The mandatory getMERMOntology operation is responsible of allowing external agents to retrieve an ontology containing the MERM. No input is needed for this operation. The result of the operation will be an ontology containing the MERM description.

A request to perform the getMERMOntology operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is

**OWL Document getMERMOntology() throws TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Mandatory | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | | | | |
| **Returns** | **Type** | | **Description** | |
| | Owl Document | | An ontology containing the MERM description | |
| **Throws** | **Type** | | **Cause** | |
| | TaTooInternalError | | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. | |

**Table 4-13: Specification of the getMERMOntology operation**

#### 4.1.4.3 Specification of the listDomains operation

The mandatory listDomains operation provides to external agents a list of domains loaded into the knowledge base. No input is needed for this operation. The result of the operation will be a set of URIs identifying the different domains.

A request to perform the listDomains operation shall include the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signatures of the operation is

**Set <URI> listDomains() throws TaTooInternalError**

| **Overrides** | Not applicable | | | |
|---|---|---|---|---|
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Mandatory | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | | | | |
| **Returns** | **Type** | | **Description** | |
| | Set <URI> | | A list of URIs identifying the domains loaded into the knowledge base. | |
| **Throws** | **Type** | | **Cause** | |
| | TaTooInternalError | | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. | |

**Table 4-14: Specification of the listDomains operation**

## 4.2. Evaluation Service

The TaToo Evaluation Service is a service of the TaToo system that belongs to the TaToo Public Services layer. It provides functionalities for a community-based evaluation of the TaToo resources, resource annotations, and single annotation tags. The service's operations are supposed to be invoked by the TaToo evaluation portlet, which belongs to the TaToo portal, as well as any other client application/tool that employs the TaToo services and provides a user support for the resource/annotation/tag evaluation.

### 4.2.1 Overview and Outline

This section provides an overview of the component describing the nature and the role of the component.

#### 4.2.1.1 Role and Scope of the Service

The Evaluation Service exposes publicly accessible TaToo evaluation functionality to the respective user components.

The Evaluation Service interacts with the Evaluation Portlet from which it receives the evaluation request and the Clearinghouse service through which its sends the request to the TaToo business layer, that is, to the Evaluation Processor.

The Evaluation Service is supposed to be an interoperable and standalone Web service developed according to widely adopted standards such as XML, WSDL, and SOAP.

#### 4.2.1.2 Service specification summary

The interface of the Evaluation Service provides the following operations:

- *addResourceEvaluation*, adds an evaluation to a resource;
- *addAnnotationEvaluation*, adds an evaluation to a resource annotation;
- *addTagEvaluation,* adds an evaluation to an annotation tag
- *getResourceEvaluations,* retrieves all evaluations of a given resource;
- *getAnnotationEvaluations,* retrieves all evaluations of a given annotation;
- *getTagEvaluations,* retrieves all evaluations of a given tag;

Please note that the *addTagEvaluation* and *getTagEvaluations* operations are optional, since such a fine granularity of evaluations is currently not needed nor considered during the discovery process.

### 4.2.2 Context

This section describes the relationships between the Evaluation Service and its context, including applied standards, relationships with other TaToo components and the TaToo resource model.

#### 4.2.2.1 Relations to standards

The Evaluation Service will be a Web service, whose design follows W3C recommendations. It will be a SOAP-based Web Service exposing a WSDL interface that provides the given set of operations. A brief description of WSDL and REST can be found in chapter 3, referenced standards.

#### 4.2.2.2 Relations to other TaToo components

The Evaluation Service is accessible via its interface from Evaluation Tools, specifically the Evaluation Portlet or other portlets and third party applications.

 

The service interacts with the Clearinghouse service to access the TaToo business layer functionalities (i.e., Evaluation Processor) that are responsible to generate and store resource, annotation and tag evaluations to the TaToo RDF repository. Moreover, the service also interacts with the Clearinghouse service to retrieve evaluations of the resources, annotations, and tags.

#### 4.2.2.3 Relations to information models

The Evaluation Service's request must contain information necessary to generate the TaToo resource, annotation and tag evaluations as they are defined in MERM. The MERM ontology provides concepts and properties that define all of the three types of the TaToo evaluations.

More information about MERM can be found in D3.1.1 - Semantic Service Environment and Framework Architecture, chapter 6.4 - Minimum Environmental Resource Model.

### 4.2.3 Specification of the Evaluation Service Interface

This section provides the functional specification of the Evaluation Service, including the service's interface and its operations. Table 4-2 provides a list of the service's operations.

| Operation Name | Description |
|---|---|
| addResourceEvaluation | Mandatory operation that adds an evaluation to the resource. |
| addAnnotationEvaluation | Mandatory operation that adds an evaluation to the resource annotation. |
| addTagEvaluation | Optional operation that adds an evaluation to the annotation tag. |
| getResourceEvaluations | Mandatory operation that returns a list of the resource's evaluations. |
| getAnnotationEvaluations | Mandatory operation that returns a list of the annotation's evaluations. |
| getTagEvaluations | Optional operation that returns a list of the tag's evaluations. |

**Table 4-2: Tagging Service Interface**

#### 4.2.3.1 Specification of the addResourceEvaluation operation

This operation calls its counterpart "addResourceEvaluation" operation of the Clearinghouse service and forwards the evaluation request to it.

A request to perform the *addResourceEvaluation* operation includes the parameters listed and defined in the table below. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter.

Furthermore the "Description" shall state the consequences for service instances if the correspondent parameter is optional and omitted.

The signature of the operation is:

**String addResourceEvaluation (resourceURI, evaluatorID, evaluationCriterion, evaluationValue, languageID) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Madatory | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | resourceURI | String | Mandatory | URI of the resource |
| | evaluatorID | String | Mandatory | ID of the evaluator |
| | evaluationCriterion | String | Mandatory | Evaluation criteria |
| | evaluationValue | int | Mandatory | Evaluation value |
| | languageID | String | Optional | Language |
| **Returns** | **Type** | | **Description** | |
| | String | | Status indicator and the URI of the newly added evaluation | |
| **Throws** | **Type** | | **Cause** | |
| | InvalidParameterValue | | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. | |
| | MissingParameterValue | | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. | |
| | ResourcesNotFound | | The resources identified by the provided URIs could not be found. | |

| | | |
|---|---|---|
| | TaTooInternalError | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. |

**Table 4-21: Specification of the addResourceEvaluation Operation**

#### 4.2.3.2 Specification of the addAnnotationEvaluation operation

This operation calls its counterpart "*addAnnotationEvaluation*" operation of the Clearinghouse service and forwards the evaluation request to it.

The signature of the operation is:

**String addAnnotationEvaluation(annotationURI, evaluatorID, evaluationCriterion, evaluationValue, languageID) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Manadatory | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | annotationURI | String | Mandatory | URI of the annotation |
| | evaluatorID | String | Mandatory | ID of the evaluator |
| | evaluationCriterion | String | Mandatory | Evaluation criteria |
| | evaluationValue | int | Mandatory | Evaluation value |
| | languageID | String | Optional | Language |
| **Returns** | **Type** | | **Description** | |
| | String | | Status indicator and the URI of the newly added evaluation | |
| **Throws** | **Type** | | **Cause** | |
| | InvalidParameterValue | | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. | |

| | | |
|---|---|---|
| | MissingParameterValue | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. |
| | ResourcesNotFound | The resources identified by the provided URIs could not be found. |
| | TaTooInternalError | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. |

**Table 4-22: Specification of the addAnnotationEvaluation Operation**

#### 4.2.3.3 Specification of the addTagEvaluation operation

This optional operation calls its counterpart "*addTagEvaluation*" operation of the Clearinghouse (if available) and forwards the evaluation request to it.

The signature of the operation is:

**String addTagEvaluation(annotationURI, tagProperty, tagValue, evaluatorID, evaluationCriterion, evaluationValue, languageID) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Optional | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | annotationURI | String | Mandatory | URI of the annotation |
| | tagProperty | String | Mandatory | Tag's property |
| | tagValue | String | Mandatory | Tag's value |
| | evaluatorID | String | Mandatory | ID of the evaluator |
| | evaluationCriterion | String | Mandatory | Evaluation criteria |
| | evaluationValue | int | Mandatory | Evaluation value |
| | languageID | String | Optional | Language |
| **Returns** | **Type** | | **Description** | |
| | String | | Status indicator and the URI of the newly added evaluation | |

| Throws | Type | Cause |
|---|---|---|
| | InvalidParameterValue | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. |
| | MissingParameterValue | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. |
| | ResourcesNotFound | The resources identified by the provided URIs could not be found. |
| | TaTooInternalError | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. |

**Table 4-23: Specification of the addTagEvaluation Operation**

#### 4.2.3.4 Specification of the getResourceEvaluations operation

This operation calls its counterpart "*getResourceEvaluations*" operation of the Clearinghouse service and forwards the evaluation request to it.

The signature of the operation is:

**List<Evaluation> getResourceEvaluations(resourceURI) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| Overrides | Not applicable | | | |
|---|---|---|---|---|
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Mandatory | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | resourceURI | String | Mandatory | URI of the resource |
| **Returns** | **Type** | | **Description** | |

| | | |
|---|---|---|
| | List<Evaluation> | List of evaluations in a form of pairs (evaluation criterion, evaluation value) |
| **Throws** | **Type** | **Cause** |
| | InvalidParameterValue | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. |
| | MissingParameterValue | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. |
| | ResourcesNotFound | The resources identified by the provided URIs could not be found. |
| | TaTooInternalError | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. |

**Table 4-24: Specification of the getResourceEvaluation Operation**

#### 4.2.3.5 Specification of the getAnnotationEvaluations operation

This operation calls its counterpart "*getAnnotationEvaluations*" operation of the Clearinghouse service and forwards the evaluation request to it.

The signature of the operation is:

**List<Evaluation> getAnnotationEvaluations (annotationURI) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| | | | | |
|---|---|---|---|---|
| **Overrides** | Not applicable | | | |
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Mandatory | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |
| | annotationURI | String | Mandatory | URI of the annotation |

| Returns | Type | Description |
|---|---|---|
| | List<Evaluation> | List of evaluations in a form of pairs (evaluation criterion, evaluation value) |
| **Throws** | **Type** | **Cause** |
| | InvalidParameterValue | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. |
| | MissingParameterValue | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. |
| | ResourcesNotFound | The resources identified by the provided URIs could not be found. |
| | TaTooInternalError | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. |

**Table 4-25: Specification of the getAnnotationEvaluations Operation**

#### 4.2.3.6 Specification of the getTagEvaluations operation

This optional operation calls its counterpart "*addTagEvaluations*" operation of the Clearinghouse (if available) and forwards the evaluation request to it.

The signature of the operation is:

**List<Evaluation> getTagEvaluations(annotationURI, tagProperty, tagValue) throws InvalidParameterValue, MissingParameterValue, ResourcesNotFound, TaTooInternalError**

| Overrides | Not applicable | | | |
|---|---|---|---|---|
| **Preconditions** | None | | | |
| **Post conditions** | None | | | |
| **Use** | Optional | | | |
| **Receives** | **Name** | **Type** | **Use** | **Description** |

| | | | | |
|---|---|---|---|---|
| | annotationURI | String | Mandatory | URI of the annotation |
| | tagProperty | String | Mandatory | Tag's property |
| | tagValue | String | Mandatory | Tag's value |
| **Returns** | **Type** | | **Description** | |
| | List<Evaluation> | | List of evaluations in a form of pairs (evaluation criterion, evaluation value) | |
| **Throws** | **Type** | | **Cause** | |
| | InvalidParameterValue | | Operation request contains an invalid parameter value (e.g. out of range, malformed document, etc.). Returns the name of the parameter with invalid value. | |
| | MissingParameterValue | | Operation request either does not include a parameter value or a empty list or map. Returns the name of the missing parameter. | |
| | ResourcesNotFound | | The resources identified by the provided URIs could not be found. | |
| | TaTooInternalError | | A problem occurred during the processing of the request by a TaToo Core Component, e.g. a specific processor. Returns the internal exception thrown by the respective Core Component. | |

**Table 4-26: Specification of the getTagEvaluations Operation**

# 5. Core Components

The Tagging Processors are a set of TaToo Core Components taking part in the tagging process and supporting functionality to the Clearinghouse. In principle these components can provide a wide range of functionality. The concrete identification of these components has been achieved for the first implementation phase resulting in the identification of two components: a component for converting formats / schemas (The Schema Mapping component) and a component for supporting the visualisation and the filtering of tags (the Visualisation & Filtering component). For the second implementation phase, the RDF Tagger Component and Evaluation Processor were identified.

## 5.1. Schema Mapping Component

This section defines the functional specification of the Schema Mapping Component and is based on the Template for the Specification of TaToo Components (TaToo-D312b). It includes an overview of the role of the component in the landscape of TaToo, a description of the relations of the component with its context and a specification of the objectives and functionality of the component.

### 5.1.1 Overview and outline

This section provides an overview of the component describing the Nature and the Role of the component.

#### 5.1.1.1 Nature of the component

The Schema Mapping Component is a special kind of a Tagging Processor, and thus belongs to the TaToo Core Component Building Block.

The Schema Mapping Component supports the mapping from one xml-schema to another where the mapping rules are described in XSLT.

#### 5.1.1.2 Role and scope of the component

The Schema Mapping Component is a component that can be used for the mapping of different meta-information schemas in the TaToo tagging process. It provides functionality that is related to the mapping of tags from a source into a target schema. We consider a schema mapping to be "the definition of an automated transformation of each instance of a data structure A into an instance of a data structure B that preserves the intended meaning of the original information" (Doerr, 2004). Thus, during the mapping process the semantics of the information are preserved.

The Schema Mapping Component is a generic component that can map between different schemas as long as:

- the information to be mapped is encoded in XML or an XML-based dialect (e.g. RDF);
- the target and source schemas are described in the XML-Schema Language; and
- an XSL document containing the schema mapping rules exists.

The description of the schema mapping is required as an input, whereby the client is the responsible for providing the appropriate mapping rules. It is outside the scope of the Schema Mapping Component to automatically derive a mapping between two schemas.

The usability of the Schema Mapping Component in the TaToo tagging process depends both on the compliance of the input schema to TaToo's Minimum Environmental Resource Model (MERM) as well as on the quality of the mapping rules. Due to the limited expressiveness of XSLT tags encoded as simple triples or property-value lists without any information about the structure of annotations are therefore are not supported by the Schema Mapping Component.

### 5.1.2 Context

This section describes the relationships between the component and its context, including technical requirements, other TaToo Components, etc.

#### 5.1.2.1 Relation to technical requirements

The Schema Mapping Component addresses the following technical requirements as specified in D2.3.2 – Requirements Document V2 (TaToo-D232). The technical requirements are mapped to the concrete functional requirements specified in chapter 5.1.3.3.

| Requirements ID and Name | Scope | Functional Requirement | Comments |
|---|---|---|---|
| TR.TAGGING.000 - Tagging means | S/I | 1 | currently no special requirements regarding schema mapping exist. |

**Table 5-1: Schema Mapping Component technical requirements**

#### 5.1.2.2 Relations to standards

The Schema Mapping Component uses XSLT for the schema transformation and thus supports various xml-dialects like, for example, RDF. A brief description of XSLT and RDF can be found in chapter 2.

#### 5.1.2.3 Relations to other TaToo components

The Schema Mapping Component can be invoked by the Tagging Service through the Clearinghouse to perform a mapping (tag conversion). It can only be used to perform a transformation; it does not store annotations as ontology instances in the knowledgebase.

Please note, that in V2 the Tagging Service does not support encoding formats for tags other than RDF. Therefore Schema Mapping Component is currently not used during the tagging process.

#### 5.1.2.4 Relations to information models

The Schema Mapping Component processes information that relate to a certain meta-information schema that is encoded in an xml-based dialect as for example RDF.

### 5.1.3 Specification

This section includes the functional specification of the component, including objectives, functional requirements, etc.

#### 5.1.3.1 Objectives

The objective of the Schema Mapping Component is to map meta-information (tags) from a source into a target schema. It takes meta-information (e.g. an RDF document) in XML format and a description of the mapping from the source to the target schema in the XSLT language as input and returns the meta-information in the target schema.

#### 5.1.3.2 Target users

The Schema Mapping Component is intended to be used by the Clearinghouse to perform requests done by the TaToo Tagging Service.

#### 5.1.3.3 Functional requirements

The functional requirement for the Schema Mapping Component is:

1. Schema Mapping: Map from one schema into another.

#### 5.1.3.4 Interaction with other TaToo components

The Schema Mapping Component will interact with the Clearinghouse to perform requests by the Tagging Service.

 

## 5.2. Visualisation & Filtering Component

This section defines the functional specification of the Visualisation & Filtering Component and is based on the Template for the Specification of TaToo Components. It includes an overview of the role of the component in the landscape of TaToo, a description of the relations of the component with its context and a specification of the objectives and functionality of the component.

### 5.2.1 Overview and outline

This section provides an overview of the component describing the Nature and the Role of the component.

#### 5.2.1.1 Nature of the component

The Visualisation & Filtering Component is a special kind of a Tagging Processor, and thus belongs to the TaToo Core Component Building Block.

In V1 the Visualisation & Filtering Component supported simple visualisation of tags and filtering of resources based on tags. In V2 the visualisation capabilities were extended and filtering is possible for users and groups as well.

#### 5.2.1.2 Role and scope of the component

The Visualisation & Filtering Component is the component responsible to provide visualisation of tags and tags filtering capabilities for user components.

The Visualisation & Filtering Component can be subdivided into two subcomponents:

- Tag Visualiser: Visualises tags associated to a resource in a way specified by the client (User Component) e.g. Tag cloud, tree, etc. Grouping tags, highlighting certain tags, etc.
- Tag Filter: Filters tags by specific tag values.

### 5.2.2 Context

This section describes the relationships between the component and its context, including technical requirements, other TaToo Components, etc.

 

#### 5.2.2.1 Relation to technical requirements

The Visualisation & Filtering Component addresses the following technical requirements as specified in D2.3.2 – "Requirements Document V2". The technical requirements are mapped to the concrete functional requirements specified in chapter 5.2.3.3.

| Requirements ID and Name | Scope | Functional Requirement | Comments |
|---|---|---|---|
| TR.TAGGING.000 - Tagging means | S/I | 1 | currently no special requirements regarding visualisation & filtering exist. |

**Table 5-2: Visualisation & Filtering Component technical requirements**

#### 5.2.2.2 Relations to standards

The Visualisation & Filtering Component uses RDF for the description of tags. A brief description of RDF can be found in chapter 3.

#### 5.2.2.3 Relations to other TaToo Components

The Visualisation & Filtering Component can be invoked by the Tagging Service through the Clearinghouse to perform visualisation and filtering. It depends on the Semantic Processor to retrieve the meta-information (tags in RDF-Format) from the knowledge base. For the specification of the Semantic Processor, please refer to D3.1.2 - Semantic Service Environment and Framework Architecture V2 (TaToo-D312), section 7.3.12 - Semantic Processor.

#### 5.2.2.4 Relations to information models

The Visualisation & Filtering Component processes information that relate to a certain meta-information schema that is encoded in an xml-based dialect as for example RDF.

### 5.2.3 Specification

This section includes the functional specification of the component, including objectives, functional requirements, etc.

#### 5.2.3.1 Objectives

The objective of the Visualisation & Filtering Component is to visualise tags and perform filtering operations requested by User Components. It takes meta-information (e.g. an RDF document) in XML format and user input to perform certain operations.

 

#### 5.2.3.2 Target users

The Visualisation & Filtering Component is intended to be used by the Clearinghouse to perform requests done by the TaToo Tagging Service and by the Semantic Processor to provide tagging information for semantic processing.

#### 5.2.3.3 Functional requirements

The functional requirements for the Visualisation & Filtering Component are:

- Visualisation: Visualise tags associated to resources;
- Filtering: Filter tags by specific tag values.

#### 5.2.3.4 Interaction with other TaToo components

The Visualisation & Filtering Component will interact with the TaToo Semantic Processor to retrieve meta-information information and with the Clearinghouse to perform requests by the Tagging Service.

## 5.3. RDF Tagger Component

This section defines the functional specification of the RDF Tagger Component and is based on the Template for the Specification of TaToo Components (TaToo-D312b). It includes an overview of the role of the component in the landscape of TaToo, a description of the relations of the component with its context and a specification of the objectives and functionality of the component.

### 5.3.1 Overview and outline

This section provides an overview of the component describing the Nature and the Role of the component.

#### 5.3.1.1 Nature of the component

The RDF Tagger is a special kind of a Tagging Processor, and thus belongs to the TaToo Core Component Building Block.

The RDF Tagger Component is the tagging processor responsible for storing tags in the TaToo knowledgebase represented by the Semantic Processor in a format that is compliant to the MERM ontology.

#### 5.3.1.2 Role and scope of the component

The RDF Tagger Component implements the actual business logic of the Tagging Service. Thus it provides in principle the same functionality as the Tagging Service. This includes currently functionality to

- associates annotations with resources,
- to retrieve annotations associated with a resource, and
- optionally to edit and delete annotations.

The RDF Tagger is furthermore responsible to encode simple triples (subject, predicate, object) received from the Tagging Service (through the Clearinghouse) as instances of the MERM ontology and to store them as RDF in the knowledgebase.

### 5.3.2 Context

This section describes the relationships between the component and its context, including technical requirements, other TaToo Components, etc.

#### 5.3.2.1 Relation to technical requirements

The RDF Tagger Component addresses the following technical requirements as specified in D2.3.2 – Requirements Document V2 (TaToo-D232). The technical requirements are mapped to the concrete functional requirements specified in chapter 5.1.3.3.

| Requirements ID and Name | Scope | Functional Requirement | Comments |
|---|---|---|---|
| TR.TAGGING.020 Access to tags (TaToos) | S/I | 1 | By interfacing with the Semantic Processor. |
| TR.TAGGING.060 Semantic Tags | S/I | 1, 2 | By encoding annotations tags as instances of the MERM ontology. |
| TR.TAGGING.080 Ontology supported tagging | S/I | 1, 2 | see above. |
| TR.TAGGING.100 Editing of Tags (TaToos) | S/I | 2 | Optional in V2 depending on the concept for updating / deleting annotations. |

**Table 5-3: RDF Tagger Component technical requirements**

#### 5.3.2.2 Relations to standards

The RDF Tagger Component returns the tags associated by the user to the resource to the Clearinghouse as RDF-Triples, it stores tags as RDF triples in the knowledgebase. A brief description of RDF can be found in chapter 2.

#### 5.3.2.3 Relations to other TaToo components

The RDF Tagger Component can be invoked by the Tagging Service through the Clearinghouse to store and retrieve tags from the knowledgebase. For this purpose, it interacts with the Semantic Processor.

#### 5.3.2.4 Relations to information models

The RDF Tagger Component stores tags as instances of the MERM Ontology.

### 5.3.3 Specification

This section includes the functional specification of the component, including objectives, functional requirements, etc.

#### 5.3.3.1 Objectives

The objective of the RDF Tagger Component is to store and retrieve tags that are encoded as ontology instances of the MERM ontology.

#### 5.3.3.2 Target users

The RDF Tagger Component is intended to be used by the Clearinghouse to perform requests initiated by the Tagging Service.

#### 5.3.3.3 Functional requirements

The functional requirements for the RDF Tagger Component are:

1. Convert and store tags: store tags retrieved from the tagging service as MERM compliant ontology instances in the TaToo knowledgebase (Semantic Processor). Depending on the format / encoding of tags used by the tagging service, a conversion might be needed, e.g. from property / value lists to RDF.

2. Retrieve and convert tags: retrieve tags from the TaToo knowledgebase. Depending on the format / encoding of tags used by the tagging service, a conversion might be needed, e.g. from RDF served by the knowledgebase to property / value lists.

 

#### 5.3.3.4 Interaction with other TaToo components

The RDF Tagger Component will interact with the Clearinghouse to perform requests by the Tagging Service and interact with the Semantic Repository to store and retrieve RDF triples.

## 5.4. Evaluation Processor

The TaToo Evaluation Processor is a component of the TaToo business layer. It is responsible for generating the TaToo evaluations based on the evaluation information received as part of the evaluation request.

The processor receives the evaluation request from the TaToo evaluation service and generates the TaToo evaluations according to the TaToo evaluation schema. The TaToo evaluation schema is defined as a part of the MERM ontology.

The communication between the TaToo Evaluation Service and Processor is not direct. It is realised through the Clearinghouse service that acts as a single entry point to the TaToo business layer.

### 5.4.1 Overview and Outline

This section provides an overview of the component describing the nature and the role of the component.

#### 5.4.1.1 Nature of the component

The Evaluation Processor is currently the default Processor for the Evaluation Service, and thus belongs to the TaToo Core Component Building Block.

The Evaluation Processor is supposed to be an interoperable and standalone Web service developed according to widely adopted standards such as XML, WSDL, and SOAP.

#### 5.4.1.2 Role and Scope of the component

The Evaluation Processor exposes operations that are accessible to the respective user components through the Clearinghouse.

The Evaluation Processor interacts with the Clearinghouse from which it receives the evaluation request and to which it sends the response of the invoked operation. Moreover, the processor has access to the TaToo RDF repository into which it stores generated evaluations as well as retrieves requested evaluations.

 

The Evaluation Processor implements the actual business logic of the Evaluation Service. Thus it provides in principle the same functionality as the Evaluation Service. This includes currently the following operations:

- *addResourceEvaluation*, generates and adds an evaluation to a resource;
- *addAnnotationEvaluation*, generates and adds an evaluation to a resource annotation;
- *addTagEvaluation,* generates and adds an evaluation to an annotation tag
- *getResourceEvaluations,* retrieves all evaluations of a given resource;
- *getAnnotationEvaluations,* retrieves all evaluations of a given annotation;
- *getTagEvaluations,* retrieves all evaluations of a given tag;

### 5.4.2 Context

This section describes the relationships between the Evaluation Processor and its context, including applied standards, relationships with other TaToo components and the TaToo resource model.

#### 5.4.2.1 Relation to technical requirements

The Evaluation Processor addresses the following technical requirements as specified in D2.3.2 – Requirements Document V2 (TaToo-D232). The technical requirements are mapped to the concrete functional requirements specified in chapter 5.1.3.3.

| Requirements ID and Name | Scope | Functional Requirement | Comments |
|---|---|---|---|
| TR.ARCH.050 - Evaluation of resources | S/I | 1, 4 | By providing the possibility to add evaluations to resources. |
| TR.REPR.000 - Evaluation of annotations | S/I | 2, 5 | By providing the possibility to add evaluations to annotations. |
| TR.DAQ.010 - Ranking Indicators | S/I | 1, 2, 3, 4, 5, 6 | By providing specific evaluations for rankings. |
| TR.TAGGING.140 - Tagging of Tags | S/I | 3,6 | By providing the possibility to add evaluations to tags. |

**Table 5-4: Evaluation Processor technical requirements**

#### 5.4.2.2 Relations to standards

The Evaluation Processor will be a Web service, whose design follows W3C recommendations. It will be a SOAP-based Web Service exposing a WSDL interface that provides the given set of

operations. A brief description of WSDL and REST can be found in chapter 2, referenced standards.

#### 5.4.2.3 Relations to other TaToo components

The same as the other components of the TaToo business layer, the Evaluation Processor is accessible via the Clearinghouse service. The service uses the API of the TaToo RDF repository to store the evaluations to it. It also uses the SPARQL endpoint of the repository to query the TaToo knowledge base for requested evaluations.

#### 5.4.2.4 Relations to information models

The evaluation request sent to the Evaluation Processor must contain information necessary to generate valid TaToo evaluations (i.e., resource evaluations, annotation evaluations, and tag evaluations). This information is defined in the MERM ontology, which provides concepts and properties that define all of the three types of the TaToo evaluations.

More information about MERM can be found in D3.1.2 - Semantic Service Environment and Framework Architecture V2 (TaToo-D312), chapter 6.4 - Minimum Environmental Resource Model.

### 5.4.3 Specification

This section includes the functional specification of the component, including objectives, functional requirements, etc.

#### 5.4.3.1 Objectives

The objective of the Evaluation Processor is to generate the TaToo evaluations according to the TaToo evaluation schema and to store them in the TaToo knowledgebase. The TaToo evaluation schema is defined as a part of the MERM ontology.

#### 5.4.3.2 Target users

The Evaluation Processor is intended to be used by the Clearinghouse to perform requests initiated by the Evaluation Service.

#### 5.4.3.3 Functional requirements

The functional requirements for the Evaluation Processor are:

3. Add resource evaluation: generate and add an evaluation to a whole resource, store the evaluation in the knowledgebase.

4. Add annotation evaluation: generate and add an evaluation to an annotation, store the evaluation in the knowledgebase.

1. Add tag evaluation: generate and add an evaluation to a single tag, store the evaluation in the knowledgebase.

2. Get resource evaluations: retrieves all evaluations of a given resource.

3. Get annotation evaluations: retrieve all evaluations of a given annotation.

4. Get tag evaluations: retrieve all evaluations of a given tag.

#### 5.4.3.4 Interaction with other TaToo components

The Evaluation Processor will interact with the Clearinghouse to perform requests by the Evaluation Service and interact with the Semantic Repository to store and retrieve evaluations.

# 6. Conclusions

The objective of this second version of TaToo Semantic Tagging Tools and Services Specifications was the specification of the components and the functionality to be implemented during the second iteration.

Components taking part in the provisioning of evaluation functionality have been presented. In particular, the Evaluation Service and the Evaluation Processor have been specified. New methods for editing and deleting tags were added to the Tagging Service in order to improve the already existing functionality and a new RDF Tagger component has been introduced.

A security layer both at client and server side has been added to provide a certain degree of authorization and authentication to the services. Improvements have been strongly influenced by the inclusion in this iteration of the multilingual and security vertical layers. As result of this inclusion, additional (optional) parameters have been included in several service operations in order to identify users and their preferred language.

The third iteration (V3) of the TaToo Semantic Tagging Tools and Services Specifications will concentrate on the further improvement of tagging and evaluation components, whereby special attention will be paid on the user interfaces of the user components. Furthermore, new functionalities, e.g. support for geospatial tagging, will be added and specifications will be updated accordingly.

 

# 7. Acknowledgements

“The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement Number 247893.”

# 8. References


**TaToo DOW, 2009** TaToo Consortium: Annex I (Description of Work) of the TaToo Grant Agreement Nr. 247893, 2009.

**TaToo-D232, 2011** Božić B., Schimak G.: Requirements document – V2, Deliverable 2.3.2 of TaToo Project, Public Document, 2011.

**TaToo-D312, 2010** Dihé P., Schlobinski S., TaToo Semantic Service Environment and Framework Architecture – V2, Deliverable 3.1.2 of TaToo Project, Public Document, 2011.

**TaToo-D312a, 2011** Dihé P., Schlobinski S., TaToo Service Specification Template, Annex of Deliverable 3.1.2 TaToo Semantic Service Environment and Framework Architecture – V2 of TaToo Project, Public Document, 2011.

**TaToo-D312b, 2011** Dihé P., Schlobinski S., TaToo Component Specification Template, Annex of Deliverable 3.1.2 TaToo Semantic Service Environment and Framework Architecture – V2 of TaToo Project, Public Document, 2011.


 