SPECIFICATION OF MODELS FOR REPRESENTING SINGLE-USER AND COMMUNITY-BASED ANNOTATIONS OF WEB RESOURCES

Tobias Burger and Olga Morozova and Ilya Zaihrayeu and Pierre Andrews and Juan Pane

January 2010

Technical Report # DISI-10-008
INSEMTIVES

Deliverable 2.1.2

Specification of Models for Representing Single-user and Community-based Annotations of Web Resources

Editor: Tobias Bürger, STI, University of Innsbruck
Deliverable nature: R
Dissemination level: Public (PU)
(Confidentiality)
Contractual delivery date: 30.09.2009
Actual delivery date: 30.09.2009
Version: 1.0
Total number of pages: 23
Keywords: semantic annotation

Abstract

INSEMTIVES overall goal is to increase the amount of available semantic content by looking into incentives and by bridging the gap between human and computational intelligence. Annotations are one form of semantic content which are supposed to make information machine readable and which typically attach additional meaning to a resource based on an implicit analysis of the resource by a human user. This deliverable builds upon an earlier deliverable which reported on a classification scheme for annotation models and an analysis of the requirements of the use cases partners in the project (cf. INSEMTIVES D2.1.1 “Report on the State-of-the-Art and Requirements for Annotation Representation Models” [3]). Based on that, it presents a consolidated annotation model and a formalization of it.
Disclaimer
This document contains material, which is the copyright of certain INSEMTIVES consortium parties, and may not be reproduced or copied without permission.

All INSEMTIVES consortium parties have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the INSEMTIVES consortium as a whole, nor a certain party of the INSEMTIVES consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum
INSEMTIVES – Incentives for Semantics

INSEMTIVES
WP2: Models and Methods for the Creation and Usage of Lightweight, Structured Knowledge
Specification of Models for Representing Single-user and Community-based Annotations of Web Resources
[Editor: Tobias Bürger, STI, University of Innsbruck]
[Work-package leader: Ilya Zaihrayeu, University of Trento]

Copyright notice
©2009-2012 Participants in project INSEMTIVES

Acknowledgement
The project is co-funded by the European Union, through the ICT Cooperation programme
Executive Summary

INSEMTIVES overall goal is to bridge the gap between human and computational intelligence in the current semantic content authoring R&D landscape in order to increase the amount and quality of annotations for a broad range of different resource types. As such it will develop methodologies for the creation of semantic content which intelligently combine automated approaches with human tasks. In order to foster user participation it will investigate incentive mechanisms to motivate users to contribute semantic annotations.

Work package 2, ”Models and Methods for Creation of Lightweight, Structured Knowledge”, develops models for representing semantic contents and their provenance, explicitly and implicitly generated by users. Apart from this, the work package develops a set of methods necessary to support the full life cycle of these semantic contents, starting from their generation and continuing with their maintenance and evolution in time.

This deliverable builds upon the deliverable D2.1.1 “Report on the State-of-the-Art and Requirements for Annotation Representation Models” which reported on a classification scheme for annotation models and an analysis of the requirements of the use cases partners in the project. The classification scheme groups annotation models according to their most prevalent features such as structural complexity, vocabulary type used, or collaboration support. The main results of the successor deliverable was an annotation model to be applied in the project. This deliverable presents a formalization of it, that is, the consolidated requirements summarized into a “generic model” and the use case extensions to this model.

We summarize the requirements briefly in this document. Based on that we present their formalization using a mathematical notation. To conclude the deliverable we outline several implementation possibilities of the model based on database and semantic technology. As such this deliverable lays the foundation for the upcoming uptake and implementation of the annotation model in the project.
# List of Authors

<table>
<thead>
<tr>
<th>Company</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>University of Innsbruck</td>
<td>Tobias Bürger</td>
</tr>
<tr>
<td>University of Innsbruck</td>
<td>Olga Morozova</td>
</tr>
<tr>
<td>University of Trento</td>
<td>Ilya Zaihrayeu</td>
</tr>
<tr>
<td>University of Trento</td>
<td>Pierre Andrews</td>
</tr>
<tr>
<td>University of Trento</td>
<td>Juan Pane</td>
</tr>
</tbody>
</table>
Contents

Executive Summary 3
List of Authors 4
Abbreviations 8
Definitions 9

1 Introduction 10
   Motivation .................................................. 10
   Scope ......................................................... 10

2 Requirements for Annotation Representation Models 11

3 The INSEMTIVES Annotation Model 13
   3.1 Purpose ..................................................... 13
   3.2 Formalization .............................................. 13
      3.2.1 Generic Model ....................................... 13
      3.2.2 Use Case Extensions ............................... 17

4 Model Implementation Guidelines 19
   4.1 Utility Functions .......................................... 19
   4.2 Implementation Technologies ......................... 19
      4.2.1 Example using Database Technology ............... 19
      4.2.2 Example Using Semantic Web Technology .......... 19

5 Conclusions and Outlook 22

References 23
**List of Figures**

1. Core Elements of the Generic Model ........................................... 13
2. Annotation Elements Defined in the Generic Model ............................ 14
3. Controlled Vocabularies Defined in the Generic Model .......................... 15
4. Controlled Annotation - Elements ...................................................... 16
5. Uncontrolled Annotation - Elements ................................................... 16
6. Access control .................................................................................. 17
7. Granularity ...................................................................................... 18
8. Example of a Model Implementation using Relational Database Technology ........................................... 20
9. Example of a Model Implementation using Semantic Web Technology .......... 20
List of Tables

1. Summary of the Requirements for each Use Case [3] .................................................. 12
Abbreviations

**OWL**  Web Ontology Language

**RDF**  Resource Description Framework

**URI**  Unified Resource Identifier

**XML**  Extensible Markup Language
Definitions

The following definitions have been partially duplicated from the definitions in deliverable D2.1.1:

**Annotation** The term *annotation* is used both as a noun denoting a piece of additional information and as a verb referring to the process of creating this additional information.

**Annotation Model** An *annotation model* defines the actual form in which the annotation, that is, additional information is expressed, and how it is linked to the original content being annotated.

**Attribute** An *attribute annotation element* is a pair \(\langle AN, AV \rangle\), where \(AN\) is the name of the attribute and \(AV\) is the value of the attribute [11].

**Authority File** An *authority file* is a kind of controlled vocabulary. In an authority file synonymous terms are grouped into concepts and one of the terms is selected as the concept name and used for visualization and navigation purposes (this term is called the *preferred term*) [5].

**Controlled Vocabulary** *Controlled vocabularies* are organized lists of words and phrases, or notation systems, that are used to initially tag content, and then to find it through navigation or search [10].

**Ontology** As defined by Studer et al., “an ontology is an explicit specification of a (shared) conceptualization” [9]

**Relation** A relation annotation element is a pair \(\langle Rel, Res \rangle\), where \(Rel\) is the name of the relation and \(Res\) is another resource (i.e., different from the one being annotated).

**Resource** A *resource* is described by an annotation within annotation models. A resource can be any identifiable content identified by an unique identifier such as URIs. Examples of resources in the context of the Web are electronic documents such as images or text documents or their parts (e.g., a reference to an entity from a text document or an area in an image).

**Semantic Annotation** *Semantic annotation* describes both the process and the resulting annotation or metadata consisting of aligning a resource or a part of it with a description of some of its properties and characteristics with respect to a formal conceptual model or ontology [8].

**Tag** A *tag* annotation element is a non-hierarchical keyword or free-from term assigned to a resource [12].

**Taxonomy** A *taxonomy* is an extension of the authority files controlled vocabulary with the broader term (BT) and narrower term (NT) relations which are defined between controlled vocabulary concepts [5, 7].

**User** A *user* in the context of this deliverable provides annotations for resources.
1 Introduction

Motivation

As identified earlier in the project, annotations which make features of content explicit, are a must to enable reliable retrieval of foremost audiovisual content. To consolidate different annotation needs and the diversity of existing annotation models, either informal or formal ones, INSEMTIVES set out to provide a generic model for representing annotations and their features. The model is supposed to be adaptable to support the implementation of various scenarios.

In the course of an earlier deliverable we analyzed possible scenarios and annotation tools relevant for the project and provided prose descriptions of extracted requirements. In order to be implementable, a more formal and structured presentation of these requirements is necessary. Therefore this deliverable provides a formalization of the INSEMTIVES annotation model and its features. Furthermore it sketches possibilities and gives recommendations for the implementation of the model and its use case extensions such as the usage of semantic technologies (e.g., RDF and OWL).

The readers of this deliverable should be familiar with the following deliverable: D2.1.1 “Report on the State-of-the-Art and Requirements for Annotation Representation Models” (cf. [3]).

Scope

This deliverable has two main objectives: first to describe the INSEMTIVES annotation model using a mathematical notation. The second objective is to demonstrate implementation possibilities.
2 Requirements for Annotation Representation Models

As previously mentioned, this deliverable is the successor of deliverable D2.1.1 (“Report on the State-of-the-Art and Requirements for Annotation Representation Models”) [3] in which requirements for annotation representation models were gathered and formulated. This section summarizes the results of the previous deliverable and gives an overview of the annotation model proposed in it.

The requirements have been elaborated with the use case partners in the project using structured interviews. The requirements gathering phase resulted in a generic model together with use case specific extensions: The core concept of the generic annotation model defined in the deliverable is a resource which is described by an annotation provided by a subject (human or not). A resource in the context of INSEMTIVES is any digital artifact identifiable with an unique identifier.

The classification scheme for annotation models developed in D2.1.1 groups existing schemas according to the level of their formality and structural richness and the required level of user involvement in the annotation process. The annotation features have been grouped into the following categories:

- **Structural Complexity**: Here we distinguish between *tags*, *attributes*, *relations*, *ontologies*, and the granularity of the annotation (i.e., *parts* or *whole*)
- **Vocabulary Type**: The different types of vocabularies supported can be *free text*, *authority file*, or *taxonomy*. Further we distinguish between different roles, i.e., who provides the vocabularies such as experts, expert-users, or end-users.
- **Provenance & Versioning**: These features represent the need to capture the history of annotations.
- **Access**: Here we distinguish between *read* or *write* access to annotations.

The possible values for the features together with a summary of the requirements for each use case is provided in Table 1. More details about the requirements are provided in [3] and are summarized in the following:

**Telefonica’s use case** will provide an annotation platform for users of an Intranet. The users will be able to annotate blog entries and forum posts in HTML format but will also be able to annotate news videos. For all type of resources, Telefonica requires the possibility to annotate parts of the resource with tags, attributed annotations or relations to other resources. These annotations will mostly come from a controlled vocabulary that will be built bottom up in the envisioned scenario.

**PGP’s use case** will provide an annotation platform for users of an online virtual world called MyTinyPlanets. An aspect to be highlighted for the PGP use case is that it consists of three sub-use cases, each covering different types of content and target groups. As for TID, the annotation complexity is mixed but the vocabulary will come from a pool of experts (the system administrators).

**SEEKDA’s use case** will provide annotation tools for the annotation of Web services. Again, this will require the annotation of attributes, tags and relations to other resources. Part of the vocabulary used will be controlled by a pool of experts while some annotations will be left open to the user’s vocabulary.

A set of generic requirements have been set to provide a minimal platform that is supposed to support all the use cases with minimal extensions.

Based on the developed requirements, the subsequent sections provide a formalization (cf. Section 3) and implementation guidelines for the model (cf. Section 4.2).
<table>
<thead>
<tr>
<th>Read Write</th>
<th>Provenance Versioning</th>
<th>Freetext API Taxonomy Provider</th>
<th>Vocabulary Type</th>
<th>Structural Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>everyone</td>
<td>experts</td>
<td>parts</td>
<td>parts</td>
<td></td>
</tr>
<tr>
<td>everyone</td>
<td>experts</td>
<td>whole</td>
<td>whole</td>
<td></td>
</tr>
<tr>
<td>controlled</td>
<td>experts</td>
<td>^</td>
<td>^</td>
<td></td>
</tr>
<tr>
<td>controlled</td>
<td>experts+users</td>
<td>^</td>
<td>^</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Summary of the Requirements for each Use Case [3]

For SEEKDA, the free-text annotations are independent from the controlled lexicon.
In the Telefonica use case, the free-text annotations will be mapped to a controlled lexicon.
3 The INSEMTIVES Annotation Model

3.1 Purpose

The purpose of the annotation model developed in the INSEMTIVES project is to capture explicit and implicit formal annotation which describe a resource or a part of a resource. The interested reader is referred to an in-depth discussion of the model in [3].

3.2 Formalization

In this section we present the formalization of the Insemtives annotation model to fit the requirements specified in the deliverable D2.1.1 (cf. [3]).

We start by introducing a generic model that can be used for every use case defined in the project (cf. [4, 2, 6]). In the next section, we present a set of extensions to this generic model: access control, annotation granularity and annotation versioning to be able to match the requirements of the specific use cases.

3.2.1 Generic Model

The generic model defines the following generic objects: user, resource, and term. Their relationship is visualized in Figure 1 and their descriptions are provided in the following.

![Figure 1: Core Elements of the Generic Model](image)

**GENERIC OBJECTS**

**Model Object 1 (User)**
A user $u$ is a tuple $u = \langle id, name \rangle$, where $id$ is a unique identifier of the user; and $name$ is the name of the user. We write $U$ to denote the set of all users.

**Model Object 2 (Resource)**
A resource $r$ is a tuple $r = \langle id, name \rangle$, where $id$ is a unique identifier of the resource; and $name$ is the name of the resource. We write $R$ to denote the set of all resources.

**Model Object 3 (Term)**
A term $t$ is a non-empty finite sequence of characters. We write $T$ to denote the set of all possible terms.

Normally, terms represent natural language words such as “sea”, “bird”, or ”location”.
CONTROLLED VOCABULARIES

The generic model defines a set of controlled vocabularies to restrict the meaning of terms describing a resource:

Model Object 4 (Synset)

A synset $syn$ is a tuple $syn = \langle sid, ST, pt, desc \rangle$, where $sid$ is a unique identifier of the synset; $ST$ is a set of synonymous terms ($ST \subseteq T$); $pt$ is a term from $ST$ (i.e., $pt \in ST$) which is called the preferred term of the synset; and $desc$ is a natural language description of the meaning of the synonymous terms.

The set of terms $ST$ models the fact that the same concept may be referred to with several synonymous words (e.g., “image” and “picture”). Preferred term serves rather for presentational purposes and is used, for example, as a synset name in the user interface. The description $d$ serves to help a human user understand the meaning of the synonymous terms (e.g., “a visual representation (of an object or scene or person or abstraction) produced on a surface”).

Model Object 5 (Natural Language Dictionary)

A natural language dictionary $nld$ is a tuple $nld = \langle nlid, LT, SYN \rangle$, where $nlid$ is a unique identifier of the dictionary language; $LT$ is a set of terms of this language ($LT \subseteq T$); and $SYN$ is a set of synsets, such that the set of terms $ST$ of any synset in $SYN$ is a subset of the set of terms of the language, i.e., $ST \subseteq LT$.

Hereinafter we use the notion of concept as it is defined in description logics [1]. We write $c$ to denote a concept and we write $C$ to denote the set of all concepts.

Model Object 6 (Taxonomy)

A taxonomy $tx$ is a rooted tree where each node is a concept $c \in C$ and each parent concept $pc$ is more general than any of its child concept $cc$, i.e., $pc \sqsubseteq cc$. We write $TX$ to denote the set of all taxonomies. We write $c \in tx$ to mean that concept $c$ belongs to the set of nodes of $tx$.

Note that the term computation and concept computation functions can be used to map from natural language terms to the concepts in a taxonomy and vice versa.

ANNOTATION ELEMENTS

In the following definitions, the user element $u$ is made optional to model the situation in which an unregistered user annotates a resource. The annotation elements defined in the generic model are visualized in Figure 2. The controlled and uncontrolled annotation types are visualized in more detail in Figures 4, 5, and 3.

Figure 2: Annotation Elements Defined in the Generic Model

Model Object 7 (Uncontrolled tag annotation)

An uncontrolled tag annotation $tag_u$ is a tuple $tag_u = \langle t, r, [u, |ts, |, \alpha] \rangle$, where $t$ is the tag term used for the annotation ($t \in T$); $r$ is the annotated resource ($r \in R$); when present, $u$ is the user who created this annotation; and $ts$ is the time stamp recorded when the annotation was created. When the annotation is added
by an automatic or semi-automatic algorithm, \( \alpha \) is provided to indicate the accuracy of the algorithm that created the annotation.

**Model Object 8 (Concept Mapping Function)**

Concept mapping function \( cmf \) is a function that takes a concept and a language identifier as input and returns a synset as output, i.e., \( cmf(c, nlid) \rightarrow syn \), s.t. \( syn \) belongs to the natural language \( nld \) dictionary with identifier \( nlid \).

The concept mapping function serves for the computation of natural language representation of set-theoretic concepts.

**Model Object 9 (Linguistic Concept)**

A linguistic concept \( lc \) is a tuple \( lc = \langle c, nlid, ct \rangle \), where \( c \) is a concept (\( c \in C \)); \( nlid \) is the identifier of a natural language dictionary; and \( ct \) is a term that belongs to the set of terms of the synset that is mapped to \( c \) and \( nlid \) by the concept mapping function, i.e., \( ct \in cmf(c, nlid) \).

**Model Object 10 (Controlled tag annotation)**

A controlled tag annotation \( tag^c \) is a tuple \( tag^c = \langle lc, r, [u, ts[, \alpha] \rangle \), where \( lc \) is a linguistic concept whose concept \( c \) belongs to a taxonomy (i.e., \( c \in tx \)); \( r \) is the annotated resource (\( r \in R \)); when present, \( u \) is the user who created this annotation; and \( ts \) is the time stamp recorded when the annotation was created. When the annotation is added by an automatic or semi-automatic algorithm, \( \alpha \) is provided to indicate the accuracy of the algorithm that created the annotation.

**Model Object 11 (Uncontrolled attribute annotation)**

An uncontrolled attribute annotation \( attr^u \) is a tuple \( attr^u = \langle an, av, r, [u, ts[, \alpha] \rangle \), where \( an \) is a term denoting the attribute name (\( an \in T \)); \( av \) is the attribute value which can belong to any of the primitive data types (e.g., dates, floats, strings); \( r \) is the annotated resource (\( r \in R \)); when present, \( u \) is the user who created this annotation; and \( ts \) is the time stamp recorded when the annotation was created. When the annotation is added by an automatic or semi-automatic algorithm, \( \alpha \) is provided to indicate the accuracy of the algorithm that created the annotation.

**Model Object 12 (Controlled attribute annotation)**

A controlled attribute annotation \( attr^c \) is a tuple \( attr^c = \langle can, av, r, [u, ts[, \alpha] \rangle \), where \( can \) is a linguistic concept denoting the attribute name whose concept \( c \) belongs to a taxonomy (i.e., \( c \in tx \)); \( av \) is the attribute value which can belong to any of the primitive data types (e.g., dates, floats, strings) or which can be a linguistic concept whose concept \( c \) belongs to a taxonomy (i.e., \( c \in tx \)); \( r \) is the annotated resource (\( r \in R \)); when present, \( u \) is the user who created this annotation; and \( ts \) is the time stamp recorded when the annotation was created. When the annotation is added by an automatic or semi-automatic algorithm, \( \alpha \) is provided to indicate the accuracy of the algorithm that created the annotation.
Model Object 13 (Uncontrolled relation annotation)
An uncontrolled relation annotation $rel^u$ is a tuple $rel^u = (sr, tr, rel, [u,] ts, \alpha)$, where $sr$ is the source resource (i.e., the resource being annotated); $tr$ is the target resource (i.e., the resource used as an annotation object); $rel$ is a term that denotes the name of the relation that exist between $sr$ and $tr$ ($rel \in T$); when present, $u$ is the user who created this annotation; and $ts$ is the time stamp recorded when the annotation was created. When the annotation is added by an automatic or semi-automatic algorithm, $\alpha$ is provided to indicate the accuracy of the algorithm that created the annotation.

Model Object 14 (Controlled relation annotation)
A controlled relation annotation $rel^c$ is a tuple $rel^c = (sr, tr, crel, [u,] ts, \alpha)$, where $sr$ is the source resource (i.e., the resource being annotated); $tr$ is the target resource (i.e., the resource used as an annotation object); $crel$ is a linguistic concept that denotes the relation that exists between $sr$ and $tr$ ($crel \in tx$) and whose concept $c$ belongs to a taxonomy (i.e., $c \in tx$); when present, $u$ is the user who created this annotation; and $ts$ is the time stamp recorded when the annotation was created. When the annotation is added by an automatic or semi-automatic algorithm, $\alpha$ is provided to indicate the accuracy of the algorithm that created the annotation.
3.2.2 Use Case Extensions

ACCESS CONTROL

The TID and SEEKDA use cases require an access control extension to manage the read and write permissions on the resources. The access control extension is visualized in Figure 6.

Figure 6: Access control

Model Object 15 (User Group)
A user group \( g \) is a tuple \( g = (id, name) \), where \( id \) is a unique identifier of the group; and \( name \) is the name of the group. We write \( G \) to denote the set of all user groups.

Model Object 16 (Group Membership)
A group membership \( gm \) is a tuple \( gm = (u, g) \), where \( u \) is a registered user of the system such as \( u \in U \) as defined in 1 and \( g \) is a user group to which this user belongs such as \( g \in G \) as defined in model object 15.

Model Object 17 (Access Rule)
An access rule is a relation between a resource and a group defining the type of access right this group has on the resource. It is represented as a triple \( ac = (r, g, right) \) where \( r \in R \) as defined in model object 2 and \( g \in G \) as defined by model object 15. \( right \in \{ \text{read, write} \} \).

A user \( u \) has access to a resource \( r \) if there exist an \( ac \) for this resource with a group to which the user belongs.

GRANULARITY

The TID and PGP use cases require a more granular annotation of resources where also parts of the resource (e.g. a paragraph in a textual document, a part of an image) can be described by an annotation. For this purpose, we extend the resource definition given in 2 to be able to refer to parts of resources.

Model Object 18 (Resource)
A resource \( r \) is a triple \( r = (id, name, part) \), where \( id \) is a unique identifier of the resource; and \( name \) is the name of the resource. \( part \) is an optional descriptor that identifies the part of the full resource identified by \( name \).
The actual definition of a part is implementation dependent and will depend on the specific requirements of each use case and resource type. A part can take multiple form:

**Model Object 19** (Textual Segment)
A textual segment is part of a textual document and is defined by a tuple: \( part^t = (beginning, end) \) where \( beginning \) and \( end \) represent the number of characters from the beginning of the document to the start of the segment and the end of the segment, respectively.

**Model Object 20** (Image Segment)
A image segment is part of an image and is defined by a singleton: \( part^i = (region) \) where \( region \) denotes a geometrical region of the image.

**Model Object 21** (Video Segment)
A video segment is part of a video and is defined by a triple: \( part^v = (beginning, end, region) \) where \( beginning \) and \( end \) represent the number of seconds from the beginning of the video to the start of the segment and the end of the segment, respectively. \( region \) denotes a static geometrical region of the video during this segment of time.

The definitions of regions and segments is kept generic in order to be able to specify them more precisely for the needs of the respective use case partners.

The granularity extension is visualized in Figure 7.

**VERSIONING**

In the generic model, the annotation elements have a provenance information that stores the user and time of addition of an annotation to a resource. In the TID and SEEKDA use cases, this provenance information must be extended to a full versioning system that allows to store when the annotation was added or removed or when its value has been updated, and by whom.

To store this information, we introduce an history object to the model.

**Model Object 22** (History Element)
An history element defined as a triple: \( he = (action, ts, u, \alpha) \). Where \( action \) is the type of action that was performed on the annotation \( action \in \{ added, removed, updated \} \), \( ts \) is the timestamp when this operation was performed and \( u \in U \) is the user that performed such action. When the annotation is added by an automatic or semi-automatic algorithm, \( \alpha \) is provided to indicate the accuracy of the algorithm that created the annotation.

Each annotation element then refers to a set of history elements containing the modifications that specific annotation received. Thus, we modify the generic definitions off the Model Objects 7, 10, 11, 12, 13 and 14 to replace the \([u, ts, \alpha]\) provenance information by a set of history elements \( \{he\} \).

For instance, the Model Object 7 becomes:

**Model Object 23** (Uncontrolled tag annotation)
An uncontrolled tag annotation \( tag^u \) is a tuple \( tag^u = (t, r, \{he\}) \), where \( t \) is the tag term used for the annotation \((t \in T)\); \( r \) is the annotated resource \((r \in R)\); where \( \{he\} \) is a set of history elements storing the different versions of this annotation element.
4 Model Implementation Guidelines

This section provides a set of useful tips and guidelines for the implementation of the models described in Section 3. Particularly, Section 4.1 presents several utility methods for mapping between some elements of the model; and Section 4.2 shows how the models can be implemented using three different technologies.

4.1 Utility Functions

Model Object 24 (Term Computation Function)

Term computation function $tcf$ is a function that takes a concept and a language identifier as input and returns a term as output, i.e., $tcf(c, nlid) \rightarrow t$.

The term computation function can be implemented by following these two steps:

1. Compute a synset $syn$ for $c$ by using the concept mapping function, i.e., compute $syn = cmf(c, nlid)$; and

2. Compute term $t$ by selecting one from the set of terms $ST$ of the computed $syn$. The term $t$ may be computed by selecting the preferred term of the synset or by asking the user in the user interface.

Model Object 25 (Synset Mapping Function)

Synset mapping function $smf$ is a function that takes a synset as input and returns a concept as output, i.e., $smf(syn) \rightarrow c$.

Model Object 26 (Concept Computation Function)

Concept computation function $ccf$ is a function that takes a term $t$ and a language identifier $nlid$ as input and returns a concept as output, i.e., $ccf(t, nlid) \rightarrow c$.

The concept computation function can be implemented by following these two steps:

1. Compute a synset $syn$ for $t$ (possibly, by asking the user in the user interface). Note that $t$ must belong to the set of terms $ST$ of synset $syn$; and $syn$ must belong to the set of synsets $SYN$ of the natural language dictionary $nld$ with identifier $nlid$; and

2. Compute the corresponding concept $c$ by using the synset mapping function, i.e., compute $c = smf(syn)$.

4.2 Implementation Technologies

The purpose of this section is to provide guidelines on how the model described in Section 3 can be implemented using three different technologies. We provide the guidelines by showing minimal but sufficient examples of possible implementations.

4.2.1 Example using Database Technology

Consider the following elements of the model: user $u = \langle id, name \rangle$, resource $r = \langle id, name \rangle$, and uncontrolled tag annotation $tag_u = \langle t, r, [u, ts] \rangle$. These elements can be implemented using the relational database technology in the schema presented in Figure 8.¹

4.2.2 Example Using Semantic Web Technology

Please again consider the following elements of the model: user $u = \langle id, name \rangle$, resource $r = \langle id, name \rangle$, and uncontrolled tag annotation $tag_u = \langle t, r, [u, ts] \rangle$. These elements can be implemented using a small ontology using RDF and OWL technology as depicted in Figure 9.

The whole ontology and sample instances are available in the following listing:²

¹The schema is designed following the Crow’s Foot notation; see http://en.wikipedia.org/wiki/Entity-relationship_diagram#Crow.27s_Foot.
²The listing using the N3 notation, cf. http://www.w3.org/DesignIssues/Notation3
Figure 8: Example of a Model Implementation using Relational Database Technology

```ttl
@prefix : <#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<> a owl:Ontology;
:Resource a owl:Class;
  rdf:comment ""xsd:string;
  rdfs:label "Resource"xsd:string;
  rdfs:subClassOf owl:Thing .

:Tag a owl:Class;
  rdf:comment ""xsd:string;
  rdfs:label "Tag"xsd:string;
  rdfs:subClassOf owl:Thing .

:User a owl:Class;
  rdf:comment ""xsd:string;
  rdfs:label "User"xsd:string;
  rdfs:subClassOf owl:Thing .

:annotates a owl:ObjectProperty;
  rdf:comment ""xsd:string;
  rdfs:domain :Tag;
```

Figure 9: Example of a Model Implementation using Semantic Web Technology
5 Conclusions and Outlook

This deliverable continues the work described in the successor deliverable D2.1.1 (“Report on the State-of-the-Art and Requirements for Annotation Representation Models”) by suggesting a formalization of the annotation model defined in it. This annotation model captures the needs of the use case partners in the INSEMTIVES project to represent annotations, that is, implicit or explicit descriptions added to a resource by a human or software agent. The intention of the formalization is to provide a common ground, ready to be implemented in the project using the technologies and following the guidelines given in Section 4.2.

The model will be used as a baseline for the representation and storage of annotations in INSEMTIVES and will be used in the INSEMTIVES platform and tools to be developed in the project.
References


