

# A LiquidPub Note (v.2.0): On Credit Attribution and its Imposed Restrictions on the SKO Model

Nardine Z. Osman, Jordi Sabater, Carles Sierra

January 26, 2009

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>A Basic Framework</b>	<b>4</b>
2.1	Architectural Model: the 3 Building Blocks . . . . .	4
2.2	Relationship Types . . . . .	5
2.2.1	Between Nodes of Different Building Blocks . . . . .	5
2.2.2	Amongst Nodes of the Same Building Block . . . . .	8
2.2.3	Categorising Relations . . . . .	13
<b>3</b>	<b>Formalisation and Specification</b>	<b>14</b>
3.1	Formal Definition . . . . .	14
3.2	Specification Language . . . . .	15
3.2.1	Sample LiquidPub Specification . . . . .	16
3.2.2	Sample Organisational Charter . . . . .	16
3.3	Integrity Constraints . . . . .	18
<b>4</b>	<b>Reputation Measures</b>	<b>25</b>
4.1	Reputation of an SKO . . . . .	26
4.2	Reputation of a Researcher . . . . .	27
4.3	Reputation of a Field . . . . .	29
4.4	Deduced Reputations . . . . .	29
4.4.1	By Projection . . . . .	29
4.4.2	By Aggregation . . . . .	29
<b>5</b>	<b>Propagation of Reputation</b>	<b>30</b>
5.1	Between Nodes of Different Building Blocks . . . . .	30
5.2	Amongst Nodes of the Same Building Block . . . . .	31
<b>6</b>	<b>Useful Applications</b>	<b>32</b>
<b>7</b>	<b>Conclusion</b>	<b>36</b>
<b>A</b>	<b>LiquidPub Integrity Constraints: a Formal Specification</b>	<b>36</b>

# 1 Introduction

One of the major issues of the LiquidPub paradigm is the need for credit attribution. This attribution raises questions such as: How is the quality of contributions *fairly* assessed? How do authors receive the *right* credit? And so on. This notion of *fairness* in credit attribution holds the key to the success of such a system.

However, this notion is highly abstract. For example, what does it mean for an author to be highly reputable in a scientific community? Does it depend on whether the author has published loads of papers in a given field? Or does it depend on the impact of his work in this field? And how can such impact be computed? The problem is that the perception of reputation is subjective. Researchers personalise their opinions based on their personal requirements, their personal experience, and their personal knowledge of the social relationships amongst other researchers. For example, to select my next invited speaker, will I care about the speaker's previously published work, or my interest is mostly in the speaker's current research? This is subjective. For this reason, and in the context of scientific publications, we distinguish between three different layers of information, which are presented by Figure 1.

First, there are the facts in the LiquidPub world. For example, being an author of a given scientific knowledge object (SKO) is a fact that is explicitly declared in the LiquidPub system. We refer to this information as the objective information, and we define it as that information which is made public and is verifiable.

Second, there are the organisational charters. For example, the IJCAI conference may provide a definition for a reputable reviewer, or for strong collaboration ties between two scientists. However, the WWW conference may disagree and provide its own definitions. Therefore, we refer to this information as the subjective information. We note that these subjectively perceived relations and reputations are based solely on the objective data of the LiquidPub system. Each organisation will have its definitions made public in its organisational charter. Various users can then make use of these charters to compute, for instance, a researcher's reputation.

Finally, the third layer represents the users' layer. Despite of having access to the LiquidPub system and organisational charters, users may also use additional sources of information, such as their own knowledge (possibly using their history of past experience), the Internet (by using techniques

<b>Users' Knowledge Bases</b> (defined relations and reputations, possibly using alternative info sources $\equiv$ <i>personalised information</i> )	
<b>Organisational Charters</b> (defined relations and reputation measures $\equiv$ <i>subjective information</i> )	<b>Alternative Info Sources</b> (information defined either in the physical world or the virtual world, such as the Internet)
<b>The LiquidPub System</b> (declared nodes and relations $\equiv$ <i>objective information</i> )	

Figure 1: The LiquidPub cake

such as web mining, for instance), or the physical world surrounding them (for example, by communicating with their friends and acquaintances). Note that these sources of information are placed in a black box in Figure 1 because they are outside the scope of this project. Nevertheless, using these varied sources of information, users can define their own notion (or perception) of relations and reputation measures accordingly. For example, one user might believe that two authors are very close friends, and hence deduce that they may be biased in their review of one another, leading to a low trust in the context of reviewing each other. This perceived trust is based on the user's personal beliefs. We refer to this deduced information as the personalised information, to distinguish it from the subjective information of organisational charters that are built entirely on top of the LiquidPub system's objective information.

Distinguishing between facts and perceptions is crucial for the success of credit attribution in the LiquidPub system, hence the proposed LiquidPub information layering model of Figure 1. The rest of this document will build upon this model accordingly.

The main goal of our research (at IIIA-CSIC) is to distinguish and define the reputation measures that could be useful for the LiquidPub system. To achieve this, relations between researchers, SKOs, and other scientific knowledge elements need to be well defined. Therefore, we first provide an overview of our proposed LiquidPub framework in Section 2, based on the point of view of the credit attribution module (which we also refer to as the reputation module). Section 3 provides a formalisation, along with a proposed specification language, for the proposed framework. Sections 4 and 5 present the reputation measures we believe are useful, along with rules on how reputation propagates amongst LiquidPub nodes, respectively. Section 6 suggests additional and more interesting applications for our reputation module. Finally, conclusions are drawn and future work is specified in Section 7.

## 2 A Basic Framework

The more generic, yet expressive, a framework is, the more useful it would be for a large variety of audiences (or applications). For the world of publications, we propose a simple framework built on three building blocks: the scientific knowledge objects (SKOs), the researchers, and the ontology. This categorisation is especially useful from the point of view of the reputation module. We avoid padding the system with extra rules on who can perform what action, since we believe this is generally context dependent and is the responsibility of the user; nevertheless, a minimum set of integrity constraints is needed to preserve the robustness of the system, as illustrated by Section 3.3. Section 2.1 presents the three building blocks, and Section 2.2 discusses the possible relationships amongst the nodes of these various blocks.

### 2.1 Architectural Model: the 3 Building Blocks

The three building blocks, or the three categories of nodes, are:

- **Scientific Knowledge Objects (SKO)s:** An SKO is the single most important node type in the world of publications: everything else revolves around it. Without these knowledge objects, publications cannot exist. An SKO may represent a published document, unpublished work, an entire conference proceedings, or any other form of scientific knowledge, such as a video, a picture, a note, a review, etc. As a result, a network of SKOs would form the main building block in the LiquidPub framework. We suggest the network representation instead of a tree simply because not all SKOs can be presented in one single tree; nevertheless, hierarchal structures will naturally exist amongst SKOs (e.g. one SKO can be a chapter in a parent SKO). Section 2.2.2 elaborates further on the various relations amongst SKOs.
- **Researchers:** SKOs cannot exist without researchers. Researchers represent the people creating, modifying, and maintaining these SKOs. Hence, along with the SKO network, a researchers network is also needed. Again, hierarchal structures may exist amongst some researchers, but a social network representation is chosen. This is because the entire set of researchers cannot be expressed in terms of one hierarchal structure; furthermore, the reputation module needs to identify these social relations, as illustrated by Section 2.2.2.

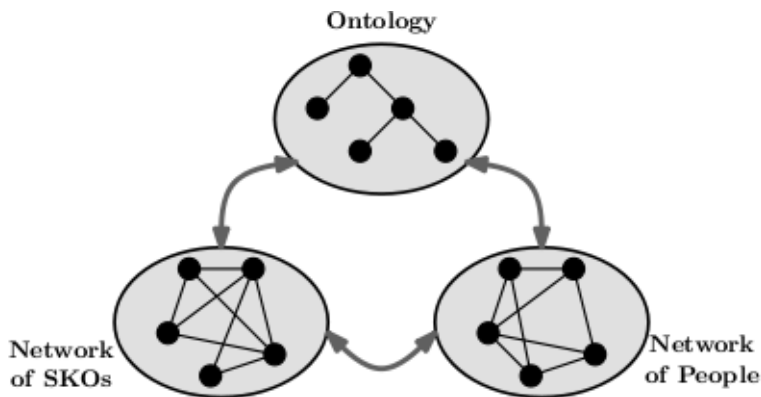


Figure 2: The LiquidPub architectural model

- **Ontology:** Last, but not least, we propose to construct a tree of ontological terms. This allows us to represent the different fields that researchers or SKOs can be linked to, such as sciences, arts, mathematics, multiagent systems, etc. Unlike SKOs and researchers, which are represented via networks, ontological terms are represented via a tree. This is because the relations amongst various fields may entirely be expressed via the sub-sumption relationship, and the root node of the ontological tree would be the “subject” (or “field” of study).

As illustrated by Figure 2, these three networks form the basic building blocks of our proposed LiquidPub framework, which we refer to as the SRO architectural model (for **S**KOs, **R**esearchers, and **O**ntology). These basic building blocks are strongly connected together. The following section discusses the various types of relationships connecting the nodes of these blocks.

## 2.2 Relationship Types

### 2.2.1 Between Nodes of Different Building Blocks

Different types of relationships may exist between researchers, SKOs, and terms of the ontology. Recall that relationships may fall into three different categories: objective, subjective, or personalised relationships (Section 1, Figure 1). In what follows, we examine each of these relationship types and the categories they fall into.

**Relations Between Researchers and SKOs** Relationship types between researchers and SKOs are based on the actions researchers are permitted to perform on SKOs, which we define as follows:

- *Owner of*: This describes the owner of the SKO, which is commonly referred to as the author. SKOs may have multiple owners. As illustrated earlier, SKOs may fall into different categories. For example, some SKOs may represent a classical document, a conference series, reviews of other SKOs, and so on. Hence, although commonly referred to as the *author*, the *owner of* relation can have various semantics, based on the category the SKO falls into. We list these below:
  - *Author*: If the SKO contains traditional knowledge content, such as text, figures, etc., then the owner is perceived as an *author*. This information may currently be automatically extracted from a document’s content. In the LiquidPub system, such information will be explicitly declared upon the creation of the SKO.
  - *Reviewer*: If the SKO contains a review of another, then the owner of the review is in fact playing the role of a *reviewer*, rather than an *author* in the classical sense. A reviewer is usually assigned by a PC member of a journal/conference/workshop/etc. Such information may currently be obtained from conference/workshop management tools, such as ConfMaster <sup>1</sup> or EasyChair <sup>2</sup>. In the LiquidPub system, this information will be explicitly declared upon the creation of the SKO.
  - *Commentator*: Similar to a reviewer, if the SKO contains a comment on another, then the owner of the comment is in fact playing the role of a *commentator*. Note that anyone in the system is allowed to comment on any visible SKO; however, only PC members may assign reviewers to professionally review an SKO.
  - *Chair*: If the SKO represents a conference, then the owner of the SKO is best described as the conference *chair*.
  - *President*: If the SKO represents a conference series, then the owner is best represented as the *president* of the conference series.

---

<sup>1</sup><http://www.confmaster.net/>

<sup>2</sup><http://www.easychair.org/>

- *Publisher of*: This describes whether the SKO has been published by some publisher registered as a LiquidPub system user.
- *Viewed by*: This describes whether the researcher has viewed (or downloaded) the SKO. Such relations can be used to measure the degree of consumption of an SKO.

Note that all of the above relations should be explicitly declared upon the creation of the SKO in question; hence, we categorise these relations as objective ones. In addition to all of the above explicitly declared relations, many other relationship types may currently be inferred simply by inspecting relations between SKOs and other SKOs. For example, if one researcher is the owner of a review that has been written on another SKO, then the researcher is also considered to have *reviewed* the latter SKO. Similarly, the *commented on* relation may be deduced by investigating the SKO comments created by one user and the other SKOs they link to. Such deduced relations would also be categorised as objective, since they are indirectly declared and verifiable.

**Relations Between SKOs and Ontological Terms** An SKO's content usually addresses a problem in a given field of study. We refer to this relation, which links an SKO to an ontological term, as the *on field* relation. Currently, this information may automatically be obtained either from the field of the journal, conference, or workshop the paper has been published in, or from the *Categories*, *General Terms*, or *Keywords* sections of the paper, if available. In the LiquidPub system, this information will be explicitly declared by the author upon the creation of its SKO. Hence, we categorise the *on field* relation as an objective one.

**Relations Between Researchers and Ontological Terms** Similarly, researchers may also be related to fields of study. Such relations may describe the knowledge (or expertise) of a scientist in a given field. But how can this information be obtained? We believe there are at least two different means for deducing these relations:

- The relation between a researcher and a given field may be deduced from the relation between the researcher's SKOs and the fields they relate to. We assume that if a researcher has produced a sufficient

number of reputable SKOs in a given field, then it is safe to conclude that the researcher is knowledgeable in that field.

- We believe it might also be possible to deduce whether a given researcher is knowledgeable in a given field by analysing the reviews he has performed in the past. We say that if a researcher has reviewed a sufficient number of SKOs in that field, then he is most probably knowledgeable enough in that field.

Whether the second method may be used on its own is a debatable issue. However, the combination of the above two methods, as opposed to using any of these methods separately, can provide a more precise measure of a researcher's knowledge in a field.

As for categorising such relations, note that a notion such as 'sufficient number of reputable SKOs' is a subjective notion. Hence, relations linking researchers to terms of the ontology, such as the *knowledgeable about*, *experienced in*, or *expert in* relations, should also be subjective. Different organisational charters will provide different definitions for these relations. Users can either use the available subjective definitions of organisational charters or specify their own personalised definition.

### 2.2.2 Amongst Nodes of the Same Building Block

In addition to the relations that could exist between researchers, SKOs, and terms of the ontology, different relationship types may also exist amongst nodes of the same type. The remainder of this section discusses these relationship types and the different categories they fall into.

**Relations Amongst Ontological Terms** The ontology is represented via a hierarchic tree of terms. Therefore, the only relationship type that we consider in this context is the sub-sumption relationship. This describes whether a given field is a parent/child of another, which we refer to as the *branch of* and *father of* relations. For example, one would say that Physics is a branch of Science while Science is the father of Physics. Currently, different organisations define different ontologies. For example, UNESCO proposes an international standard nomenclature for fields of science and technology <sup>3</sup>. Similarly, ACM defines its own classification system for the

---

<sup>3</sup><http://unesdoc.unesco.org/images/0008/000829/082946EB.pdf>

computing field <sup>4</sup>. In the LiquidPub system, all organisations should agree on one ontology, and this ontology should be modified with time as new fields are introduced. However, for the time being, we decide to explicitly declare the ontology at the initialisation stage, possibly using one of the commonly used ones, such as UNESCO's. Relations amongst terms of the ontology would therefore be objective ones.

**Relations Amongst SKOs** As we have noted earlier, hierarchal structures may exist amongst SKOs. These are used to depict the structural composition of an SKO. For example, one SKO may represent a chapter, a section, or even a figure of some parent SKO. To describe this subsumption relationship, we use the *part of* and *parent of* relations.

However, not all SKOs relate to each other in a hierarchal manner. One SKO may be an alternative version of another. This happens when authors modify their SKOs, which automatically creates new versions. We refer to this as the *version of* relation. An SKO may also be a variation of another. For example, the same work may be published in different conferences, workshops, or journals. Each of these SKOs would present the same concepts, possibly presented in a different light, with different applications and examples, or with new results. We refer to this as the *variation of* relation. Additionally, one SKO may extend the work of another, reusing old content and/or concepts. We refer to this as the *extension of* relation. Note that the *version of*, *variation of*, and *extension of* semantically describe the temporal progress of a given concept/research, as opposed to the structural composition of an SKO.

In addition to structural and temporal relations, there are also the meta relations that link two SKOs, where one would contain meta-content describing the other. These are the *review of* and *comment on* relations that allow one SKO to provide information about another. Semantically, both relations have the same meaning. The difference, however, lies in the right to declare these relations. While anyone can create a comment, only a reviewer assigned by a PC member may create a professional review.

All of the relations described above on linking SKOs together would be explicitly declared upon the creation of one of the SKOs. Hence, we categorise these relations as objective ones.

We note that many other relationship types may currently be inferred

---

<sup>4</sup><http://www.acm.org/about/class/>

simply by inspecting relations between SKOs and researchers or SKOs and ontological terms. For example, whether two SKOs are written by the same author is deduced through the inspection of the existing SKO-researcher relations. Similarly, whether two SKOs are written on the same field of study is deduced through the inspection of the existing SKO-ontology relations. These relations would also be categorised as objective, since they are indirectly declared and verifiable.

**Relations Amongst Researchers** Numerous types of relationships may exist amongst researchers. These could describe whether there exists coauthorship work between two researchers, whether the two researchers are colleagues (i.e. they work on the same project or in the same research lab), whether they are friends, whether they are family members, whether they exchange music, whether there is a love-hate relationship between them, and so on. As one can see, the list of possible researcher relations can be infinite. Therefore, we propose a two-dimensional space (inspired from the work by Ashri et al. (2005)<sup>5</sup>) to describe the relations that we believe are most useful for our reputation module.

The first dimension describes whether researchers are collaborative or competitive. The second describes whether researchers are dependent or independent. Figure 3 provides an illustration of these two dimensions, along with some sample relations. For example, point **A** could describe a standard relationship between a student and a supervisor, were the student is completely dependent on the supervisor and a very strong collaborative tie exists between the two. Point **B** could describe the relationship between two scientists working on a common project; hence, some notion of dependency exists between the two, along with relatively strong collaborative ties. On the other hand, point **C** could describe the relationship between two independent competitive scientists.

We note that each of these dimensions is complete. Researchers can either be dependent or independent. We propose the use of the interval  $[0,+1]$  to represent this *depend* relation. Similarly, researchers can either be collaborative or competitive. We propose the use of the interval  $[-1,+1]$  to

---

<sup>5</sup>Ashri et al. (2005) defines the following five relationship types for the context of an e-market: trade, dependency, competition, collaboration, and tripartite relations. In our categorisation, we make use of the semantics of dependency, competition, and collaboration between researchers to produce a two-dimensional relationship model (see Figure 3).

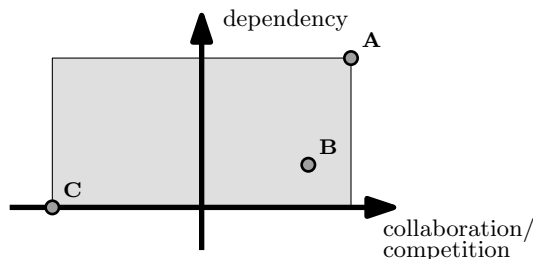


Figure 3: The 2-dimensional researchers' relationship model, demonstrating 3 different relations

represent this collaboration (or competition) relation, which we refer to as the *collab\_compete* relation. However, whether additional dimensions will be needed (or prove to be useful for our reputation module) is still an open question.

But how is the degree of these relations (*dependency/independency* and *collaboration/competition*) computed? And what kind of information is used in computing such relation measures? In what follows, we first address the latter question by providing a sample of information sources used for categorising researcher relationships. The former question is addressed subsequently. The list of possible information sources that can be used in categorising researcher relationship types is presented below:

1. Identifying whether two researchers work on the same project could possibly imply the existence of both collaborative ties and dependency between the researchers.
2. Identifying whether two researchers work in the same institute/research lab could also imply the existence of both collaborative ties and dependency between the researchers.
3. Identifying whether two researchers are on the same board could possibly imply the existence of dependency between the researchers.
4. Identifying the superior/subordinate hierarchic relations between researchers could be used to imply the existence of a strong dependency relationship (e.g. a student/supervisor relationship).
5. Identifying the strength of collaborative ties may be achieved by inspecting coauthored publications (e.g. Newman (2001) provides a method

for analysing social networks and computing the strength of collaborative ties between two scientists by taking into consideration the number of papers published by both researchers and the number of other authors, if any, on each of these papers).

6. Identifying whether reciprocity exists between two researchers (e.g. Mui (2003)) could also be used to imply the existence of collaborative ties.
7. Identifying whether two researchers are working on competitive research/projects may be deduced, for example, by identifying whether the researchers have no (or weak) collaborative ties, are both working on the same subject (or field), yet they belong to different institutes (or projects).

But what about the former question: How is the degree of relations, such as *dependency/independency* and *collaboartion/competition*, computed? We note that the information sources mentioned by points 1, 4, 6, and 7 are not available in the LiquidPub system. Therefore, relations making use of such information may only be defined by independent users making use of alternative sources of information (such as personal knowledge, or knowledge available on the web). Therefore we label the *depend* and *collab\_compete* relations making use of such information as personalised relations.

On the other hand, information sources of points 2, 3, and 5 should be declared in the LiquidPub system. For example, the LiquidPub system should allow the declaration of *coauthorship* relationships (i.e. whether two researchers coauthored the same SKO), *affiliation* relationships (i.e. whether two researchers belong to the same institution or research lab), and *same board* relationship (i.e. whether two researchers have been on the same board of a given conference). However, how this information is used to compute the intensity of dependency or collaboration between two researchers is a subjective issue, and may be defined by organisational charters. Therefore, *depend* and *collab\_compete* relations making use of this type of information are labelled as subjective relations.

### 2.2.3 Categorising Relations

As a summary to the relations presented above, Figure 4 categorises the relations discussed in the previous sections by visually categorising them into three main categories. Those in red represent subjective and personalised relations. These relations may either be defined by organisational charters or by various users. The relations in green and blue represent objective relations which are based on facts declared in the LiquidPub system. However, we differentiate between two types of objective relations: (1) those that are explicitly declared, and are highlighted with blue by Figure 4, and (2) those that are deduced from other explicitly declared information, and are highlighted with green by Figure 4. Additionally, Figure 4 also categorises the relations with respect to the node types they link: SKOs, users, and terms.

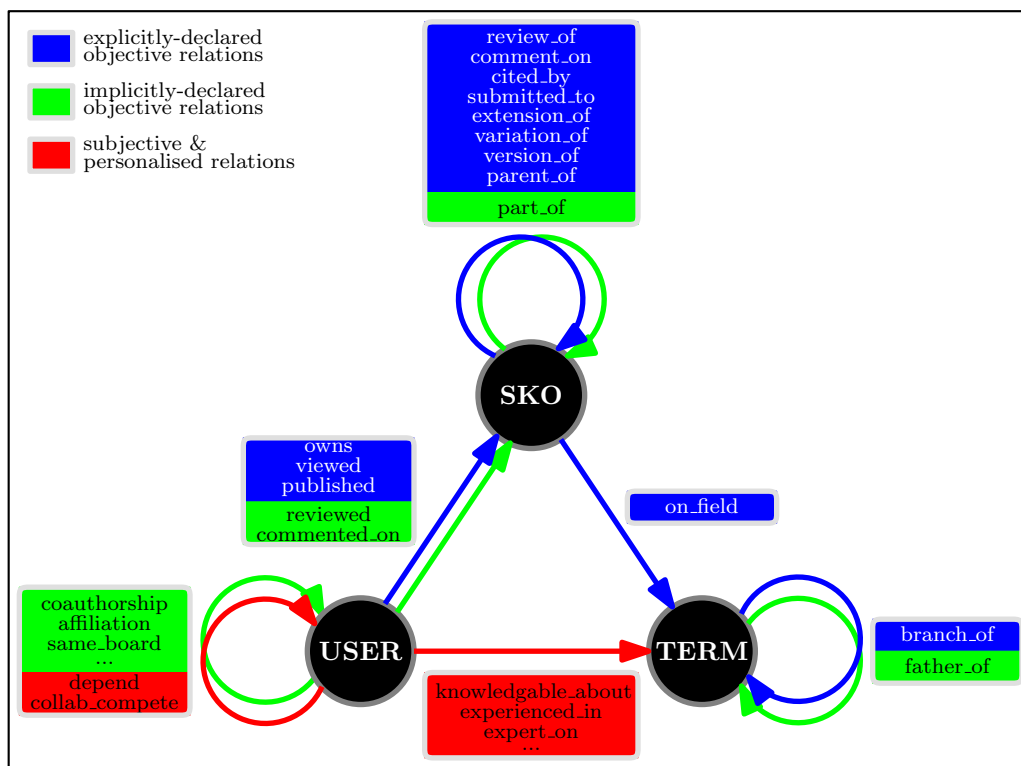


Figure 4: Categorising LiquidPub relations

### 3 Formalisation and Specification

The LiquidPub framework proposed in this document has yet to be agreed upon, formalised, and implemented. Nevertheless, we provide a formal definition of this framework in Section 3.1, along with a proposed specification language in Section 3.2. As for integrity constraints, those are discussed in Section 3.3.

#### 3.1 Formal Definition

The proposed LiquidPub framework is defined as the tuple:

$$LPF = \langle \mathcal{N}_{sko}, \mathcal{N}_{researcher}, \mathcal{N}_{field}, \mathcal{R} \rangle$$

$\mathcal{N}_{sko}$  represents the finite set of SKOs,  $\mathcal{N}_{researcher}$  represents the finite set of researchers, while  $\mathcal{N}_{field}$  represents the finite set of ontological terms, or fields of study.  $\mathcal{R}$  represents the set of various relationships that exist between nodes, which have been presented by Section 2.2. This set is currently defined as  $\mathcal{R} = \{branch\_of, father\_of, on\_field, review\_of, comment\_on, cited\_by, submitted\_to, extension\_of, variation\_of, version\_of, parent\_of, part\_of, owns, viewed, published, reviewed, commented\_on, coauthorship, affiliation, same\_board\}$ . Each relation of the set  $\mathcal{R}$  is then defined as follows:  $branch\_of \subseteq \mathcal{N}_{field} \times \mathcal{N}_{field}$ ,  $father\_of \subseteq \mathcal{N}_{field} \times \mathcal{N}_{field}$ ,  $on\_field \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{field}$ ,  $review\_of \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{sko}$ ,  $comment\_on \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{sko}$ ,  $cited\_by \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{sko}$ ,  $submitted\_to \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{sko}$ ,  $extension\_of \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{sko}$ ,  $variation\_of \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{sko}$ ,  $version\_of \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{sko}$ ,  $parent\_of \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{sko}$ ,  $part\_of \subseteq \mathcal{N}_{sko} \times \mathcal{N}_{sko}$ ,  $owns \subseteq \mathcal{N}_{researcher} \times \mathcal{N}_{sko}$ ,  $viewed \subseteq \mathcal{N}_{researcher} \times \mathcal{N}_{sko}$ ,  $published \subseteq \mathcal{N}_{researcher} \times \mathcal{N}_{sko}$ ,  $reviewed \subseteq \mathcal{N}_{researcher} \times \mathcal{N}_{sko}$ ,  $commented\_on \subseteq \mathcal{N}_{researcher} \times \mathcal{N}_{sko}$ ,  $coauthor \subseteq \mathcal{N}_{researcher} \times \mathcal{N}_{researcher}$ ,  $affiliation \subseteq \mathcal{N}_{researcher} \times \mathcal{N}_{researcher}$ ,  $same\_board \subseteq \mathcal{N}_{researcher} \times \mathcal{N}_{researcher}$ . The semantics of each relation have already been discussed by Section 2.2. We note that the set  $\mathcal{R}$  of the LiquidPub framework  $LPF$  includes only the objective relations (Figure 4). This is because the remaining relations are defined outside the LiquidPub framework. For instance, subjective relations are defined by different organisational charters, whereas personalised relations are defined by users.

## 3.2 Specification Language

As illustrated by the previous section, the LiquidPub framework is kept simple, basic, and generic. For specifying the LiquidPub framework, we propose the basic and generic language presented by Figure 5. We note that this specification language is only a sample and may be changed at a later stage. However, we present it here to give a taste of how the system may be implemented and specified in a simple and straight forward manner.

---

LPframework	::	⟨{Node,...},{Relation,...}⟩
Node	::	sko(SK0id,{Attribute,...})   researcher(RESEARCHERid,{Attribute,...})   field(FIELDid,{Attribute,...})
Relation	::	Rid   Rid :- Rdef
Rid	::	branch_of(RESEARCHERid,SK0id)   father_of(FIELDid,FIELDid)   on_field(SK0id,FIELDid)   review_of(SK0id,SK0id)   comment_on(SK0id,SK0id)   cited_by(SK0id,SK0id)   submitted_to(SK0id,SK0id)   extension_of(SK0id,SK0id)   variation_of(SK0id,SK0id)   version_of(SK0id,SK0id)   parent_of(SK0id,SK0id)   part_of(SK0id,SK0id)   owns(RESEARCHERid,SK0id)   viewed(RESEARCHERid,SK0id)   published(RESEARCHERid,SK0id)   reviewed(RESEARCHERid,SK0id)   commented_on(RESEARCHERid,SK0id)   coauthorship(RESEARCHERid, RESEARCHERid)   affiliation(RESEARCHERid, RESEARCHERid)   same_board(RESEARCHERid, RESEARCHERid)
Attribute	::	ATTid=ATTdescription
SK0id, RESEARCHERid, FIELDid, ATTid	::	ID

---

Figure 5: The LiquidPub specification language

In coherence with the formal definition of the LiquidPub framework of Section 3.1, Figure 5 states that the framework is composed of a set of nodes

(`{Node, ...}`) and a set of relations linking these nodes together (`{Relation, ...}`). Nodes in the specification language are divided into three groups: SKOs, researchers, and terms of the ontology. They are specified as: `sko(SKOId, {Attribute, ...})`, `researcher(RESEARCHERid, {Attribute, ...})`, and `field(FIELDId, {Attribute, ...})`, respectively. Similarly, the LiquidPub’s objective relations are: `branch_of`, `father_of`, `on_field`, `review_of`, `comment_on`, `cited_by`, `submitted_to`, `extension_of`, `variation_of`, `version_of`, `parent_of`, `part_of`, `owns`, `viewed`, `published`, `reviewed`, `commented_on`, `coauthorship`, `affiliation`, and `same_board`. The parameters of each relation state which nodes are linked to which others. Recall that some of these objective relations are explicitly specified in the LiquidPub system, while others are implicitly specified (Figure 4). Implicitly specified relations have definitions (`Rid :- Rdef`) that specify how the relation is deduced.

Last, but not least, we note that the `ID` in Figure 5 represents a unique identifier, while `ATTdescription` represents a term describing the value of a given attribute, and `Rdef` represents a term defining a relation.

### 3.2.1 Sample LiquidPub Specification

A sample LiquidPub system is presented by Figure 6. In this system, there are two fields of study, one author, and four SKOs. Note that some relations are explicitly specified (e.g. `branch_of`, `owns`), while others are implicitly specified, and hence, their general definition is provided (e.g. `chair`, `father_of`).

### 3.2.2 Sample Organisational Charter

As illustrated by the formal framework of Section 3.1, the LiquidPub system defines objective relations only. As for subjective relations, such as those describing the dependency and collaboration/competition between researchers, as well as those describing the expertise of a researcher in a given field, these will be defined by organisational charters. Organisational charters will build their definitions on top of the LiquidPub system, making use of its knowledge base. A sample organisational charter is presented by Figure 7.

The simplistic definition of the `knowledgable_in` relation of Figure 7 states that if a researcher is either the author or reviewer of at least one SKO in a given field, then the researcher is considered knowledgeable in that field. As for the `collab_compete` relation definition, it states that if two researchers have coauthored at least one SKO, then their collaboration measure is 1;

---

```

field(pc, [name="Process Calculi", description=""]).
field(ccs, [name="Calculus of Communicating Systems", description= ""]).

researcher(rm, [f_name="Robin", l_name="Milner", p_affiliation="Uni. of Cambridge", ...]).

sko(sko1, [title="A Calculus of Communicating Systems", type=book,
          uri="0-387-10235-3"],...).
sko(sko2, [title="Communication and Concurrency", type=book, uri="0-131-15007-3"], ...).
sko(sko3, [title="Pure Bigraphs", type=pdf,
          uri="http://www.cl.cam.ac.uk/ rm135/tutorial-7.pdf"], ...).
sko(sko4, [title="Is Informatics a Science?", type=mp3,
          uri="http://www.diffusion.ens.fr/data/audio/2007_12_10_milner.mp3"]).

branch_of(pc, ccs).

owns(sko1, rm).
owns(sko2, rm).
owns(sko3, rm).
owns(sko3, rm).

on_field(ccs, sko1).
on_field(ccs, sko2).
on_field(pc, sko3).

extension_of(sko1, sko2).

chair(SK0id, USERid) :-
    owns(SK0id, USERid),
    sko(SK0id, Attributes),
    member(sko_category="conference", Attributes).

father_of(FIELDid1, FIELDid2) :-
    branch_of(FIELDid2, FIELDid1).

...

```

---

Figure 6: A miniature example of a LiquidPub system specification

otherwise, it is 0.

Note that different charters would provide different relation definitions. Therefore, each organisational charter will be specifying (through the definitions it provides) an area in the two-dimensional researchers-relations space (see Figure 3) that it labels as strong dependency/collaboration relation.

As for system users, these can either make use of public organisation charters, or build their own relation definitions. These users are even free to create new relations as they see fit as well as make use of additional sources of information, such as their own knowledge base or information obtained by mining the web. For example, when analysing relations between

---

```

knowledgable_in(RESEARCHERid,FIELDid) :-
    author_of(RESEARCHERid,SKOid),
    on_field(SKOid,FIELDid).
knowledgable_in(RESEARCHERid,FIELDid) :-
    reviewer_of(RESEARCHERid,SKOid),
    on_field(SKOid,FIELDid).

collab_compete(RESEARCHERid1,RESEARCHERid2,1) :-
    author_of(RESEARCHERid1,SKOid),
    author_of(RESEARCHERid2,SKOid), !.
collab_compete(RESEARCHERid1,RESEARCHERid2,0).

```

---

Figure 7: An organisational charter’s sample relation definition

researchers, organisational charters have to stick with the LiquidPub declared information, such as coauthorship and affiliation information. On the other hand, users can make use of their private knowledge to decide whether two researchers are close friends, for instance.

### 3.3 Integrity Constraints

The formalisation and specification of the system have been defined in the previous sections. But what about the management of the system? We believe the system should be flexible enough to allow users to create and maintain SKOs according to their needs and contexts<sup>6</sup>. Nevertheless, the system should also be robust enough to survive the vandalism of malicious users. For example, what if a user pretends to be someone else when creating or rating an SKO? A minimum level of integrity constraints is needed to preserve the robustness of the system.

As illustrated by the previous section, users may explicitly declare nodes and relations. Therefore, we propose a set of integrity constraints to address issues revolving around the explicit declaration, modification, and deletion of nodes and relations. Note that nodes have attributes, and the content of these attributes may sometimes be relevant to some integrity constraints. Hence, we outline the proposed set of attributes for each node type in Table 1. The integrity constraints are then summarised by Tables 2, 3, and 4.

In summary, the integrity constraints of Table 2 state that field nodes may only be declared, modified, or deleted by a LiquidPub system administrator.

---

<sup>6</sup>We note that we do not plan to build editors to help users edit SKOs, but we do believe the system should allow users to use their preferred editor for downloading, modifying, and uploading an SKO.

<b>FIELD</b>	<b>RESEARCHER</b>	<b>SKO</b>
name	first_name	title
description	family_name	date
	middle_name	uri
	image	body
	primary_affiliation	type
	primary_position	category
	primary_address	access_rights
	primary_email	sko_visibility
	primary_url	comments_visibility
	secondary_affiliation	reviews_visibility
	...	

Table 1: The attributes of LiquidPub nodes

Each node should be unique, and each node should be the branch of exactly one other field node. Furthermore, a field node may be deleted only if it has no children nodes linked to it. As for researcher nodes, they may be modified and deleted by their owner only. The only constraint on researcher nodes is that the email attribute(s) should be unique and the primary email should have been verified. Note that a researcher is allowed to delete itself only if it does not own any SKOs. We also suggest the deletion of all *viewed* relations linking the researcher being deleted to the SKOs it has viewed in the past. Finally, an SKO node may only be declared by one of its owners (or authors). However, the consent of all other authors, if any, should be received to make sure coauthors agree to their access rights. For security reasons, the creation date is not set by the owner declaring the SKO, but by the LiquidPub system. Also, the LiquidPub system may automatically check for conflicts between the *uri* and *type* attributes (e.g. if the owner declares the type to be a *pdf*, but uploads a *video*). To modify an SKO, a copy of the SKO is created, and a *version\_of* relation is declared to link the newly created copy to the original SKO. An SKO may be deleted only if it has not been published. In this case, all relations linking the SKO to any other node in the system are deleted.

As for Table 3, it describes the constraints on declaring, modifying, and deleting relations (except those linking SKO nodes together, which are described by Table 4). The *branch\_of* relation linking field nodes together may only be declared or deleted upon the addition or the deletion of the corresponding field node, respectively. In addition, the LiquidPub system administrator may modify these relations to move a field node from one place in

the ontology to another. We note that it is prohibited to have ontological terms that are not the branch of any other; i.e. ontological terms should always be attached to the ontology and no free floating terms or sub-tree of terms is accepted. As for the remaining relations, modifying any of those is equivalent to deleting the old relation and declaring a new one. Hence, for simplification, we restrict the actions that may be performed on all remaining relations to the addition and deletion actions only. As illustrated by Table 3, the *on\_field* relation linking an SKO to a field node, may be added or deleted by any authorised author (i.e. an author with the appropriate access rights). Similarly, any authorised author may also add or delete an *owns* relation. However, since the *owns* relation is critical for the reputation module, the LiquidPub system needs to be sure that all owners (or coauthors) agree to the new change in the list of authors and their corresponding access rights. Additionally, every time an *on\_field* or *owns* relation is added or deleted, all other parent SKOs should have their *on\_field* and *owns* relations modified. This is because if a researcher is an author of a section of an SKO then the researcher should also be considered as one of the authors of the parent SKO. The *on\_field* relation is treated in a similar manner. As for the *published*, it may be added by the publisher only if the author has agreed to do so. Although it may be deleted by either the publisher or one of its authorised authors. Finally, the *viewed* relation may only be added by the LiquidPub system every time a researcher views, or downloads, and SKO. It may be deleted by the system only when the related researcher or SKO is deleted.

The integrity constraints on relations linking SKOs together, which are described by Table 4, state that the *parent\_of* relation may be declared by an authorised author of the parent node in two cases only. First, if consent has been received from an author of the child node. Second, if the child node has been submitted for review to the parent node. The owner of any of the child or parent nodes have the right to delete such a relation. The *version\_of* relation may only be declared or deleted by the LiquidPub system when an existing SKO has been modified or the SKO being linked to has been deleted, respectively. Only authorised authors may declare their SKOs to be a *variation\_of* or an *extension\_of* another existing one. They may also delete such relations when needed. As for the *submitted\_to* relation, if one SKO has been submitted for review to another SKO, then an authorised owner of the former SKO has the right to declare or delete this *submitted\_to* relation. In both cases, consent should be received from the list of coauthors, if any, for any submission (or cancellation of submission). The *cites* relation

may only be declared or deleted by the LiquidPub system. This, however, requires the system to be capable of extracting such information from the content of existing SKOs; although the mechanism needed for achieving this is still not clear at the moment. The author of any comment or review may create or delete a *comment\_on* or *review\_of* relation, respectively. However, the system should also check that reviews are being attached to SKOs that have been submitted to the corresponding conference (or journal, workshop, etc.).

Finally, we note that a complete formal specification of the LiquidPub integrity constraints is presented through the Z notation (Spivey, 1989) by the appendix at the end of this document.

NODE	ACTION	AUTHORISED USERS	PRE-CONDITIONS	POST-CONDITIONS
FIELD	ADD	LP administrator	<ul style="list-style-type: none"> <li>the value of the <i>name</i> attribute is unique</li> </ul>	<ul style="list-style-type: none"> <li>a <i>branch_of(name2, name)</i> relation should be declared, where <i>name2</i> is the identifier of a previously declared field and <i>name</i> is the identifier of the field being declared</li> </ul>
	MODIFY	LP administrator	<ul style="list-style-type: none"> <li>the value of the <i>name</i> attribute is unique</li> </ul>	
	DELETE	LP administrator	<ul style="list-style-type: none"> <li>no existing SKO links to the field being deleted</li> <li>no existing field is a branch of the field being deleted</li> </ul>	<ul style="list-style-type: none"> <li>all existing <i>branch_of(-, name)</i> relations should be deleted, where <i>name</i> is the identifier of the field being deleted</li> </ul>
USER	ADD	new user	<ul style="list-style-type: none"> <li>the value of the <i>primary_email</i> attribute should be unique</li> <li>the values of all the <i>secondary_email</i> attributes (if any) should be unique</li> <li>the value of the <i>primary_email</i> attribute should have been verified</li> </ul>	
	MODIFY	user being modified	<ul style="list-style-type: none"> <li>if the value of the <i>primary_email</i> attribute has been modified, then it should be unique</li> <li>if the values of any of the <i>secondary_email</i> attributes have been modified, then they should be unique</li> </ul>	
	DELETE	user being deleted	<ul style="list-style-type: none"> <li>no existing SKO is owned by this user</li> <li>no existing SKO is published by this user</li> </ul>	<ul style="list-style-type: none"> <li>all existing <i>viewed(-, primary_email)</i> relations linking SKOs to the user being deleted should also be deleted; note that <i>primary_email</i> represents the identifier of the user being deleted</li> </ul>
SKO	ADD	author	<ul style="list-style-type: none"> <li>for every author <i>c</i> of the SKO's list of coauthors <i>C</i>, receive the consent of <i>c</i> on the list of coauthors <i>C</i> and the value of the <i>access_rights</i> attribute</li> <li>the value of the <i>content_type</i> attribute does not contradict with that of the <i>uri</i></li> </ul>	<ul style="list-style-type: none"> <li>the value of the <i>date</i> attribute is set to that of the current system date</li> <li>for every author <i>c</i>, an <i>owns(uri, c)</i> relation is created linking the user <i>c</i> to the created SKO, whose identifier is <i>uri</i></li> </ul>
	MAKE COPY	authorised author	<ul style="list-style-type: none"> <li>the value of the <i>content_type</i> attribute does not contradict with that of the <i>uri</i></li> </ul>	<ul style="list-style-type: none"> <li>the value of the <i>date</i> attribute is set to that of the current system date</li> <li>a <i>version_of(uri1, uri2)</i> relation linking the SKO newly created copy (whose identifier is <i>uri2</i>) to the former copy (whose identifier is <i>uri1</i>) should be declared</li> </ul>
	DELETE	authorised author	<ul style="list-style-type: none"> <li>no existing user has published this SKO</li> </ul>	<ul style="list-style-type: none"> <li>all existing <i>on_field(-, uri)</i>, <i>viewed(uri, -)</i>, <i>owns(uri, -)</i>, <i>review_of(-, uri)</i>, <i>comment_on(-, uri)</i>, <i>cites(-, uri)</i>, <i>submitted_to(-, uri)</i>, <i>extension_of(-, uri)</i>, <i>variation_of(-, uri)</i>, <i>version_of(-, uri)</i>, and <i>parent_of(-, uri)</i> relations, linking the SKO to be deleted (whose identifier is <i>uri</i>) to other nodes, should also be deleted</li> </ul>

Table 2: LiquidPub integrity constraints governing the declaration, modification, and deletion of ndoes

RELATION	ACTION	AUTHORISED USERS	PRE-CONDITIONS	POST-CONDITIONS
<i>branch_of</i>	ADD	LP system	<ul style="list-style-type: none"> <li>this action has been invoked by the addition of a field node</li> </ul>	
	MODIFY	LP administrator	<ul style="list-style-type: none"> <li>only the first attribute of the <i>branch_of(field1, field2)</i> relation may be modified</li> </ul>	
	DELETE	LP system	<ul style="list-style-type: none"> <li>this action has been invoked by the deletion of a field node</li> </ul>	
<i>on_field</i>	ADD	authorised author		<ul style="list-style-type: none"> <li>for every parent SKO of the SKO referred to by the <i>on_field</i> relation being declared, if the parent SKO is not already related to the field linked to its child node, then declare a new <i>on_field</i> relation linking the parent SKO to this new field</li> </ul>
	DELETE	authorised author		<ul style="list-style-type: none"> <li>for every parent SKO, if the parent SKO does not have any other children SKOs related to the field referred to by the <i>on_field</i> relation being deleted, then delete all existing <i>on_field</i> relation linking the parent SKO to this field</li> </ul>
<i>owns</i>	ADD	authorised author	<ul style="list-style-type: none"> <li>the value of the <i>access_rights</i> attribute has been modified accordingly</li> <li>consent has been received from all coauthors (or co-owners) on the addition of a new author and the modified <i>access_rights</i> attribute</li> <li>receive consent from the newly created author on the modified <i>access_rights</i> attribute</li> </ul>	<ul style="list-style-type: none"> <li>for every parent SKO of the SKO referred to by the <i>owns</i> relation being declared, if the parent SKO is not already related to the researcher linked to its child node, then declare a new <i>owns</i> relation linking the parent SKO to this new researcher</li> </ul>
	DELETE	authorised author	<ul style="list-style-type: none"> <li>the remaining list of coauthors is not <math>\emptyset</math></li> <li>the value of the <i>access_rights</i> attribute has been modified accordingly</li> <li>consent has been received from all coauthors (or co-owners) on the deletion of one of them and the modification of the <i>access_rights</i> attribute</li> </ul>	<ul style="list-style-type: none"> <li>for every parent SKO, if the parent SKO does not have any other children SKOs related to the researcher referred to by the <i>owns</i> relation being deleted, then delete all existing <i>owns</i> relations linking the parent SKO to this researcher</li> </ul>
<i>published</i>	ADD	publisher	<ul style="list-style-type: none"> <li>consent has been received from the SKO's main author</li> </ul>	
	DELETE	publisher authorised author		<ul style="list-style-type: none"> <li>inform the main author of un-publishing the SKO</li> <li>inform the publisher of un-publishing the SKO</li> </ul>
<i>viewed</i>	ADD	LP system	<ul style="list-style-type: none"> <li>when a user downloads/views an SKO, a <i>viewed</i> relation is automatically declared, linking the two nodes together</li> </ul>	
	DELETE	LP system	<ul style="list-style-type: none"> <li>this action has been invoked by the deletion of one of the nodes it links together</li> </ul>	

Table 3: LiquidPub integrity constraints governing the declaration, modification, and deletion of relations

RELATION between SKO1 & SKO2	ACTION	AUTHORISED USERS	PRE-CONDITIONS	POST-CONDITIONS
<i>parent_of</i>	DECLARE	authorised author of SKO2	<ul style="list-style-type: none"> <li>consent has been received from the main author of SKO1</li> </ul>	
	DELETE	authorised author of SKO1	<ul style="list-style-type: none"> <li>SKO1 is linked to SKO2 via the <i>submitted_to</i> relation</li> </ul>	
<i>version_of</i>	DECLARE	authorised author of SKO1 or SKO2		
	DELETE	LP system	<ul style="list-style-type: none"> <li>this action has been invoked by the creation of a new copy of an existing SKO node</li> <li>this action has been invoked by the deletion of a related SKO node</li> </ul>	
<i>variation_of</i>	DECLARE	LP system		
	DELETE	LP system		
<i>extension_of</i>	DECLARE	authorised author of SKO2		
	DELETE	authorised author of SKO2		
<i>submitted_to</i>	DECLARE	authorised author of SKO2		
	DELETE	authorised author of SKO2		
<i>cites</i>	DECLARE	authorised author of SKO2	<ul style="list-style-type: none"> <li>receive consent from all coauthors for submitting the SKO for review at some organisation</li> </ul>	
	DELETE	authorised author of SKO2	<ul style="list-style-type: none"> <li>receive consent from all coauthors for canceling a submission for review at some organisation</li> </ul>	<ul style="list-style-type: none"> <li>inform the organisation that submission has been canceled</li> </ul>
<i>comment_on</i>	DECLARE	LP system		
	DELETE	LP system	<ul style="list-style-type: none"> <li>one SKO occurs in the references of another</li> </ul>	
<i>review_of</i>	DECLARE	LP system		
	DELETE	LP system	<ul style="list-style-type: none"> <li>this action has been invoked by the deletion of the related SKO node citing another node</li> </ul>	
<i>parent_of</i>	DECLARE	authorised author of SKO2		
	DELETE	authorised author of SKO2		
<i>submitted_to</i>	DECLARE	authorised author of SKO2	<ul style="list-style-type: none"> <li>If SKO1 is the review of SKO2, then there should exist a <i>submitted_to</i>(SKO3, SKO2) relation linking SKO2 to some conference SKO3, such that SKO1 is also linked to SKO3 via the <i>parent_of</i>(SKO3, SKO1) relation</li> </ul>	
	DELETE	authorised author of SKO2		

Table 4: LiquidPub integrity constraints governing the declaration, modification, and deletion of relations

## 4 Reputation Measures

Sections 2 and 3 have laid down the foundation for our work by proposing a LiquidPub framework and its formal definition, respectively. The framework is based on three interconnected basic building blocks: the SKO network, the researchers network, and the ontology. The relations connecting nodes together have also been discussed. In this section, we introduce the reputation measures that should be computed by our reputation module for the given LiquidPub framework.

As noted by Section 1, reputation is a subjective matter. Therefore, similar to subjective relations, reputation measures should also be defined by organisational charters as well as system users. However, defining reputation from scratch is a complex matter. The reputation module we propose for the LiquidPub system should provide a varied collection of basic reputation measures, which can be combined (and possibly modified) by organisational charters or system users as they see fit. The remainder of this section elaborates further on this.

Given the three node types – the SKO, the researcher, and the ontological term (or field of study) – we start by investigating how the reputation of each node type may be computed (Sections 4.1, 4.2, and 4.3, respectively). We then proceed to illustrate how further reputation measures may be deduced through projection and combination (Section 4.4). However, first and foremost, we introduce the concept of opinions, that the subsequent sections refer to.

**Reputation and Opinions** As the following sections will illustrate, reputation may be based on several sources of information. However, one most commonly used source is that of the group’s opinion. We therefore provide a brief introduction to this concept. Sierra and Debenham (2008) state that reputation is essentially the opinion of a group about something. Opinions are then presented as probability distributions over an evaluation space. For example, the evaluation space may be defined as the set {`excellent`, `very good`, `good`, `poor`, `bad`}. Opinions can then be collected and aggregated to obtain the group opinion, also known as the reputation measure.

In what follows, we use the notion `opinion(object, attribute, context)=value` to specify opinions. Note that an opinion may refer to a specific attribute of the object being evaluated. For example, one can specify that the novelty of one’s work is excellent but the clarity in explaining it is poor. Additionally,

an opinion may refer to a given context. For example, if the object being evaluated is a researcher, then the context is used to describe the different roles played by a researcher. For example, one researcher may be a very good reviewer, but a bad author. Of course, the object field is mandatory; however, the attribute and context fields are not. The value should be a probability distribution. However, for simplification, the examples used in this document specify the value as a single number in the range [0,1]. To achieve this, the evaluation space is simplified into {good, bad}. Then, instead of saying, for example, `value={good=0.7, bad=0.3}`, we simply say `value=0.7`.

With the given background on opinions and their representation, we now proceed to discuss the various reputation measures that may be computed in the LiquidPub system, several of which make use of opinions.

## 4.1 Reputation of an SKO

Several sources of information may be used to compute the reputation of an SKO. These could be:

- Pagerank style measures, which consider the number/percentage of other SKOs referencing the SKO in question
- review results, which are measures that may be obtained for SKOs that have been submitted for reviews
- ratings made by other researchers in the community, which requires a careful analysis of the reliability of each reputation measure obtained from these third-party members (Sabater and Sierra, 2002)

Our reputation module should provide different computational models for different sources of information. Since reputation is a subjective measure, each organisational charter (or each independent user) can then combine these computational models as they see fit, giving each the weight they believe it deserves for their context.

But what about computing the reputation of more challenging concepts, such as the novelty of some work, how complete an SKO is, and so on? Such measures will have to rely solely on reviewers' and researchers' opinions (the 2<sup>nd</sup> and 3<sup>rd</sup> sources of information in the list above). For example, the opinion may be specified as follows: `opinion(object, attribute)`, where `object` represents the SKO in question and `attribute` represents the attribute being rated, such as the novelty of the work, the clarity of an SKO, etc.

We note that discussing the reputation of an SKO raises a crucial question: How are sub-SKOs (e.g. chapters, sections, figures, etc.) dealt with? In a more general form, one can rephrase this question as follows: How does reputation propagate within related SKOs? Section 5 revisits this issue.

## 4.2 Reputation of a Researcher

Several sources of information may be used to compute the reputation of a researcher. These could be:

1. the researcher's degree centrality in his social network; in other words, how many other scientists is this researcher connected to?
2. the researcher's betweenness centrality in his social network; in other words, how important is the researcher in connecting other scientists together?
3. the researcher's closeness centrality in his social network; in other words, how closely connected is the researcher to all other researchers in his community?
4. the researcher's prestige degree in his social network; in other words, how much is this researcher being sought in his community? Note that this could be a measure on how often is the researcher invited to talks, invited for collaboration, or even how much is the researcher's SKOs referenced by others (Pujol et al., 2002).
5. the researcher's  $h$  index, which is a measure of the researcher's scientific productivity and impact
6. the reputation of the researcher's SKOs, which is dependent on the propagation of reputation mechanism in the LiquidPub framework, and is discussed in more detail in Section 5
7. the rating of other researchers in the community, which (again) requires a careful analysis of the reliability of each reputation measure obtained from a third-party member (Sabater and Sierra, 2002). We note that ratings might relate to various attributes (e.g. coherence, bias, etc.) and different contexts (e.g. for authors, reviewers, etc.).

8. the researcher's reviewing capabilities; in other words, how do previous review results compare to the final (or the group's) review results?
9. the researcher's relation to others; in other words, is there a dependency or collaborative/competitive relationship between researchers?
10. the researcher's history of bias; in other words, is the researcher known to be biased against a race, a gender, a nationality, a topic, a programming language, or even a methodological approach?

Again, our reputation module should provide different computational models for different sources of information. Organisational charters may then combine and weight these models as they see fit. For instance, different contexts require different combinations of the above. A general reputation of a researcher may combine all of the above. The reputation of an author might best be described by the combination of points 4-7 above. On the other hand, the reputation of a reviewer would strongly be influenced by points 8-10. As for points 1-3, these are measures of how well the researcher is connected in his community. The type of relationships used in constructing the social network affects the meaning of the researcher's degree centrality, his betweenness centrality, and his closeness centrality. If only coauthorship relations are used, then the reputation of a good collaborator may be described by the combination of points 1-7.

But what about more challenging reputation measures, such as the reputation measure of an invited speaker? We believe such measures can be based solely on reviewers' and researchers' opinions (or the seventh source of information in the previous list). For example, the opinion may be specified as follows: `opinion(object, attribute, context)`, where `object` represents the researcher in question, `attribute` represents the attribute being rated (such as novelty of research, clarity in explaining, etc.), and `context` describes the role of the researcher (for example, one researcher might be clear in explaining things on paper but not orally; hence, he would have the following two reputation measures: `opinion(researcherId, clarity, author)=0.85` and `opinion(researcherId, clarity, speaker)=0.25`).

Alternatively, one might decide that the next invited speaker should be someone, for example, who can deliver an interesting talk. In such a case, the computation of this measure will require different type of information, such as information on who shares my interest. This type of information

may be obtained by independent users by investigating who has been viewing similar SKOs to theirs, who has sufficient publications written on the fields of their interest, and so on. At the time being, it is still not clear whether such information may be easily obtained. The available literature provides examples on how this problem may be approached. Yu and Singh (2003) suggest a method for computing the similarity between researchers. Heath et al. (2007) proposes a method for computing the affinity measure between two researchers. Nevertheless, both mechanisms suggest the use of information that would be defined outside the LiquidPub system.

### 4.3 Reputation of a Field

We believe that the reputation of a given field is fully dependent on the reputation of its publications. This requires a clear definition of how reputation propagates between an ontological term and a given SKO. Section 5 revisits this issue.

### 4.4 Deduced Reputations

As illustrated by the previous sections, the main reputation measures are those of SKOs and researchers. We refer to these as the *basic reputation* measures. Other measures may be deduced by performing different operations, such as projection and combination, which we present below.

#### 4.4.1 By Projection

Many reputation measures may be deduced by applying projection. For example, to compute the reputation of a researcher in a given field, the exact same methods of computing the general reputation of the researcher will apply; however, the methods will be applied only to the researchers and SKOs related to the selected field. The reputation of an SKO in a given field may be computed in a similar manner.

#### 4.4.2 By Aggregation

As illustrated by Section 4.3, the reputation of a given field is defined as the combination of reputation measures of all SKOs in that specific field. Similarly, the reputation of a project or a research lab is also a combination

of all reputation measures of SKOs associated with the entity in question (whether it is a project, a meeting, or a research lab).

## 5 Propagation of Reputation

The previous section has presented the reputation measures that may be computed by our reputation module. However, several of these measures are obtained through the propagation of reputation amongst nodes. The propagation of reputation strongly relies on the relationship types between the nodes, which were presented earlier by Section 2.2. In what follows, we present the assumptions we propose on the propagation of reputation.

### 5.1 Between Nodes of Different Building Blocks

**Between Researchers and SKOs** A researcher’s reputation is a (direct or indirect) result of his SKOs’ reputation. Similarly, one might think that if a researcher is a reputable author then his SKOs are worth inspecting. However, it is not necessarily the case that reputable researchers only produce reputable SKOs, or that unreputable researchers only produce unreputable SKOs. Therefore, we assume the following.

**Assumption 1.** *Reputation does not propagate from a researcher to the SKOs he owns.*

**Assumption 2.** *Reputation of a researcher shall take into consideration the aggregation of the reputation of the SKOs he owns.*

We note that this aggregation should carefully consider the relations existing amongst the researcher’s SKOs. For example, a researcher who has published  $n$  reputable SKOs that are mostly linked to each other through the *version of* relation should be treated differently from a researcher who has published  $n$  reputable SKOs that are mostly not linked to each other.

**Between SKOs and Ontologies** Similarly, the reputation of an ontological term, or a field of study, is a result of its SKOs’ reputation. Again, we believe this relation is not reciprocal. Therefore, we assume the following.

**Assumption 3.** *Reputation does not propagate from an ontological term to its related SKOs.*

**Assumption 4.** *Reputation of an ontological term shall take into consideration the aggregation of the reputation of its related SKOs.*

**Between Researchers and Ontologies** It is obvious that if a researcher is related to a given field of study then the researcher should not necessarily inherit that field’s reputation. However, it is not clear yet whether reputation should propagate from researchers to fields of study (or ontological terms). For the time being, we assume that researchers are related to ontological terms only through the SKOs they write or review. Therefore, it is through the SKO that reputation propagates. As a result, we currently suffice with the following assumption.

**Assumption 5.** *Reputation does not propagate between one researcher and his related ontological terms (or fields of study).*

## 5.2 Amongst Nodes of the Same Building Block

**Within the Researchers Network** Does the relationship one researcher has with different researchers affect his reputation? For example, does the reputation of a student affect his supervisor’s, or vice versa? Does the reputation of one author affect the reputation of his collaborators? And so on. For the time being, we make this simple assumption.

**Assumption 6.** *Reputation does not propagate from one researcher to another.*

Our assumption is based on the view that a researcher’s reputation is the sole result of his hard work, and not whom he relates to. Although it is not clear yet whether this assumption holds for all contexts.

**Within the Ontology** The ontology is represented as a tree. Therefore, as illustrated by Section 2.2, we only consider the subsumption relations between ontological terms when defining the propagation of reputation within the ontological tree. The following are the assumptions we make.

**Assumption 7.** *Reputation does not propagate down a tree, i.e. from a parent node to a child node.*

**Assumption 8.** *Reputation of a parent node shall take into consideration the aggregation of the reputation of its children nodes.*

**Within the SKO Network** SKOs might either relate to each other via subsumption relationships or via other relationship types, such as *version of* and *extension of*. As a result, different assumptions are made addressing the different relationship types.

**Assumption 9.** *Reputation does not propagate between SKOs related by any relation other than the ‘parent of’ and ‘part of’ relations.*

**Assumption 10.** *Reputation of an SKO, only if not explicitly specified, may be inherited either from its children or parent node(s), only if their reputation is explicitly specified. Preference is given to inheritance from children nodes.*

**Revisiting the Issue of Integrity** Keeping in mind the integrity constraints of Section 3.3, we note that assumption 9 contributes to preserving the integrity of SKOs. For example, the *parent of* or *part of* subsumption relations are structural relations describing the structure of one main SKO. In these cases, we believe an SKO’s general reputation may propagate to the different sections, chapters, or other parts of the SKO. Similarly, the reputation of all sections may be aggregated to provide the interested user with the general reputation of the parent SKO. However, when an author (who does not necessarily have to be the original author of an SKO) writes a different version of the same SKO, the resulting SKO could have a completely different reputation. We believe this should not affect the reputation of the original SKO. Similarly, when an author extends the work of another (or even itself), we believe that the reputation of this extension does not necessarily have to affect older work.

## 6 Useful Applications

Section 4 has illustrated both the basic and deducible reputation measures that may be calculated. However, we believe a much more interesting application of the reputation module for the LiquidPub context may be achieved by assisting the automated publication process. In what follows, we discuss the various actions of a conference publication process that we believe our reputation module can assist with. Naturally, the same process may be applied to journals, workshops, etc.

**Fishing for interesting publications from the SKO network** Being capable of computing the reputation of (both published and unpublished) SKOs implies that a conference chair can make use of such a capability to search for and invite these highly reputable SKOs. Alternatively, one can also search for reputable authors and invite their new SKOs to a conference. We remind the reader that different conferences will have different definitions for an SKO’s or researcher’s reputation provided by their charters.

But what about new SKOs that have not been reviewed by the community and belong to non-famous authors? We believe it is safe to assume that for these SKOs to gain reputation, they will have to manually apply to conferences. The review results they receive, along with the reputation of the conference they get published in, can be used as an initial measure of their reputation, before getting rated by other researchers in the community.

**Selecting PC committee members from the researchers network** Computing the reputation of a researcher in reviewing or rating (both published and unpublished) SKOs may assist in deciding who to invite as a PC member. This, we believe, does not only provide a relatively strong incentive for researchers in the community to review or rate SKOs, but also to review and rate them “properly” (in other words, to take the review process more “seriously”). This is because the closer the researcher’s review result is to the group’s result, then the higher his reputation as a reviewer is; hence, the higher his probability for playing the role of a PC member is.

**Deciding how many reviewers are needed for a given paper** Unreliable review results suggest additional reviewers should review the SKO. Therefore, computing the reliability of reviews, or even predicting it, can help decide how many additional reviewers might be needed.

For example, if there is a huge difference in two researchers’ opinions, then it is highly recommended that a third reviewer should be invited. Several methods can help us compute the distances between two opinions, which we propose to model as probability distributions. The first uses the Kullback-Leibler divergence method, while the second uses the Earth Movers Distance method. Both of these methods have been discussed by Sierra et al. (2008).

Alternatively, after selecting the reviewers needed for one SKO, and given each reviewer’s review history, how often they were close to the group’s opinion, and so on, the final group opinion may be predicted, informing the chair

whether a reliable result will probably be achieved or not. If not, then additional reviewers might be invited. The joint distribution method, the yager method, or the maximum uninorm method could be used for predicting a group's opinion. These three methods have also been discussed by Sierra et al. (2008).

**Deciding who to review a given paper** The reputation of a reviewer may also assist in deciding who should review what. The reputation strongly depends on measures that have already been discussed by Section 4.2, such as possible bias, dependencies amongst reviews, etc.

**Aggregating reviewers' results** Similarly, the reputation of a reviewer may also assist in providing the reliability measure needed for aggregating reviewers' results. How reliable is the result provided by a given reviewer depends, amongst other things, on the possible bias of the result, whether the reviewer is in a competitive or collaborative relationship with the author of the SKO being reviewed, whether there exists strong dependencies amongst reviews, and so on.

**Bringing it all together** A basic and generic interaction model may be provided for automating the entire publication process. Figure 8 provides the state graph of such a sample interaction. The parts highlighted in red represent the actions that require the use of our reputation module.

Note that the interaction could be made flexible enough to accommodate the various decisions of conference chairs. For example, the chair may decide whether or not to invite reputable SKOs, whether or not to invite reviewers for a discussion, whether or not to allow authors to defend their work by replying to reviewers' comments, and so on.

The sub-interaction model specified within the dark grey box represents the parallel instances of this sub-interaction, where each instance deals with a different SKO. As for the sub-interaction models specified within the light grey boxes, these represent parallel instances run by the various reviewers of a given SKO.

Finally, note that reviewers may discuss issues infinitely often; however, an author may only reply to a reviewer's comment once. Of course, the interaction model may easily be modified to allow authors to reply to reviewers' comments more than once, if needed.

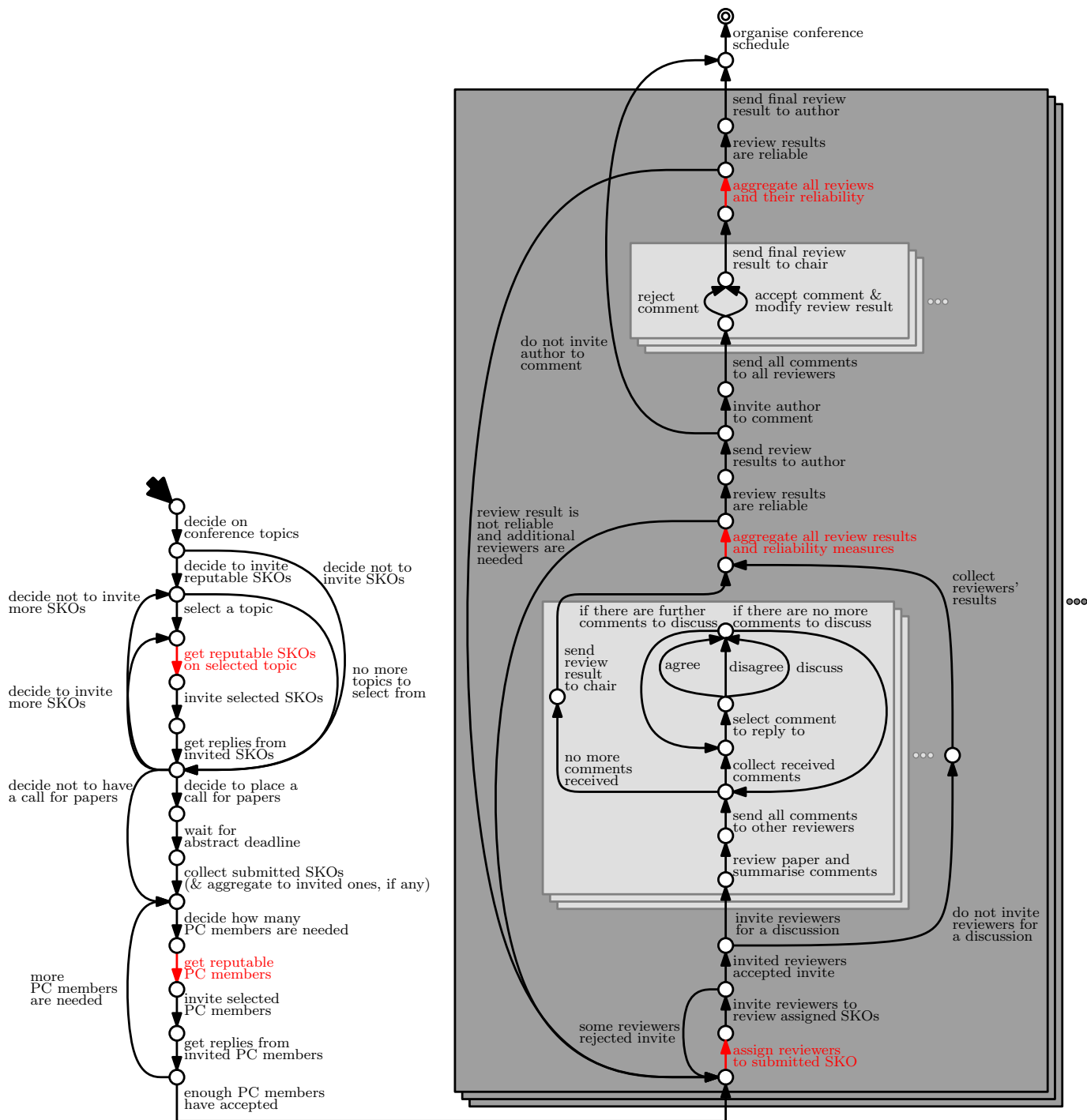


Figure 8: The state graph of an automated publication process

## 7 Conclusion

The framework presented in this document is a generic and flexible one. It divides LiquidPub nodes into three categories: SKOs, researchers, and ontological terms. The LiquidPub system should initially contain the definition of objective relations, such as *author of*, *branch of*, etc., along with the initialisation of the ontology. The system will then grow as researchers and SKOs are added with time. The addition of nodes should be a straightforward matter.

The only constraints are the integrity constraints governing the declaration, modification, and deletion of nodes and relations. These are necessary for preserving the integrity of SKO, which is crucial for assigning credit to both SKOs and their owners.

The resulting system is a flexible system. It allows organisational charters to define new relations and reputation measure, building their definitions on top of the LiquidPub system. Individual users can then make use of these public charters, or even build their own definitions making use of other sources of information, such as private knowledge.

Generic interaction models may be built to fully automate the process of preparing a conference. The automated process can make use of the the available reputation module to help conference chairs invite reputable SKOs, select reputable reviewers, etc.

## A Integrity Constraints: Formal Specification

[NAME, TEXT, EMAIL, IMAGE, POSITION, ADDRESS, URL, URI, CTYPE, STYPE, DATE, BODY, ARIGHTS, VISIBILITY]

CTYPE ::= text | html | msdoc | pdf | latex | image | video | ...

STYPE ::= default | review | comment | conference | conference\_series | ...

VISIBILITY ::= true | false

*OntTerm*

*name* :  $\mathbb{P}$  NAME

*description* : NAME  $\leftrightarrow$  TEXT

...

*User*

$primary\_email : \mathbb{P} \text{ EMAIL}$   
 $first\_name : \text{ EMAIL} \rightarrow \text{ NAME}$   
 $middle\_name : \text{ EMAIL} \leftrightarrow \text{ NAME}$   
 $last\_name : \text{ EMAIL} \rightarrow \text{ NAME}$   
 $image : \text{ EMAIL} \leftrightarrow \text{ IMAGE}$   
 $primary\_affiliation : \text{ EMAIL} \rightarrow \text{ NAME}$   
 $primary\_position : \text{ EMAIL} \rightarrow \text{ POSITION}$   
 $primary\_address : \text{ EMAIL} \rightarrow \text{ ADDRESS}$   
 $primary\_url : \text{ EMAIL} \leftrightarrow \text{ URL}$

...

...

*SKO*

$uri : \mathbb{P} \text{ URI}$   
 $date : \text{ URI} \rightarrow \text{ DATE}$   
 $title : \text{ URI} \leftrightarrow \text{ NAME}$   
 $body : \text{ URI} \leftrightarrow \text{ BODY}$   
 $content\_type : \text{ URI} \rightarrow \text{ CTYPE}$   
 $category : \text{ URI} \rightarrow \text{ STYPE}$   
 $access\_rights : \text{ URI} \rightarrow \text{ ARIGHTS}$   
 $sko\_visibility : \text{ URI} \rightarrow \text{ VISIBILITY}$   
 $downloads\_visibility : \text{ URI} \rightarrow \text{ VISIBILITY}$   
 $comments\_visibility : \text{ URI} \rightarrow \text{ VISIBILITY}$   
 $reviews\_visibility : \text{ URI} \rightarrow \text{ VISIBILITY}$   
 $submitted\_to\_review\_visibility : \text{ URI} \rightarrow \text{ VISIBILITY}$

$\forall u \in uri \bullet \exists u \rightarrow ct \in content\_type \wedge conflict(u, ct) = \perp$

*InitOntTerm*

*OntTerm*

$name = \emptyset$

*InitUser*

*User*

$primary\_email = \emptyset$

*InitSKO*

*SKO*

$uri = \emptyset$

*Relations*

$branch\_of : OntTerm \rightarrow OntTerm$

$on\_field : SKO \rightarrow OntTerm$

$owns : User \rightarrow SKO$

$downloaded : User \rightarrow SKO$

$published : User \rightarrow SKO$

$parent\_of : SKO \rightarrow SKO$

$version\_of : SKO \rightarrow SKO$

$variation\_of : SKO \rightarrow SKO$

$extension\_of : SKO \rightarrow SKO$

$submitted\_to : SKO \rightarrow SKO$

$cites : SKO \rightarrow SKO$

$comments\_on : SKO \rightarrow SKO$

$review\_of : SKO \rightarrow SKO$

*InitRelations*

*Relations*

$branch\_of = \emptyset$

$on\_field = \emptyset$

$owns = \emptyset$

$downloaded = \emptyset$

$published = \emptyset$

$parent\_of = \emptyset$

$version\_of = \emptyset$

$variation\_of = \emptyset$

$extension\_of = \emptyset$

$submitted\_to = \emptyset$

$cites = \emptyset$

$comments\_on = \emptyset$

$review\_of = \emptyset$

*AddOntTerm*

$\Delta OntTerm$

$\Delta Relations$

$name? : NAME$

$description? : NAME$

$father\_node? : NAME$

$name? \notin name \wedge$

$name' = name \cup \{name?\} \wedge$

$description? \neq \emptyset \Rightarrow description' = description \cup \{name? \leftrightarrow description?\} \wedge$

$Relationss.branch\_of' = Relationss.branch\_of \cup \{name? \rightarrow father\_node?\}$

*ModifyOntTerm*

$\Delta OntTerm$

$nameOld? : NAME$

$nameNew? : NAME$

$description? : NAME$

$(nameNew? = nameOld? \vee nameNew? \notin name) \wedge$

$name' = (name - \{nameOld?\}) \cup \{nameNew?\} \wedge$

$description' = description - \{nameOld? \leftrightarrow X\} \wedge$

$description? \neq \emptyset \Rightarrow description' = description \cup \{name? \leftrightarrow description?\}$

*DeleteOntTerm*

$\Delta OntTerm$

$\Delta Relations$

$name? : NAME$

$\nexists sko \rightarrow field \in Relations.on\_field \bullet field.name = name? \wedge$

$\nexists field1 \rightarrow field2 \in Relations.branch\_of \bullet field2.name = name? \wedge$

$name' = name - \{name?\} \wedge$

$description' = description - \{name? \leftrightarrow y\} \wedge$

$\forall rel \in Relations.branch\_of \bullet rel = field \rightarrow x \wedge field.name = name? \wedge$

$Relations.branch\_of' = Relations.branch\_of - \{rel\}$

*AddUser*

$\Delta User$

*email?* : *EMAIL*

*f\_name?* : *NAME*

*m\_name?* : *NAME*

*l\_name?* : *NAME*

*image?* : *IMAGE*

*p\_affiliation?* : *NAME*

*p\_position?* : *POSITION*

*p\_address?* : *ADDRESS*

*p\_url?* : *URL*

$email? \notin primary\_email \wedge$

$email\_confirmed(email?) \wedge$

$primary\_email' = primary\_email \cup \{email?\} \wedge$

$first\_name' = first\_name \cup \{email? \rightarrow f\_name?\} \wedge$

$m\_name? \neq \emptyset \Rightarrow middle\_name' = middle\_name \cup \{email? \rightarrow m\_name?\} \wedge$

$last\_name' = last\_name \cup \{email? \rightarrow l\_name?\} \wedge$

$image? \neq \emptyset \Rightarrow image' = image \cup \{email? \rightarrow image?\} \wedge$

$primary\_affiliation' = primary\_affiliation \cup \{email? \rightarrow p\_affiliation?\} \wedge$

$primary\_position' = primary\_position \cup \{email? \rightarrow p\_position?\} \wedge$

$primary\_address' = primary\_address \cup \{email? \rightarrow p\_address?\} \wedge$

$p\_url? \neq \emptyset \Rightarrow primary\_url' = primary\_url \cup \{email? \rightarrow p\_url?\} \wedge$

*ModifyUser*

$\Delta User$

*emailNew?* : EMAIL

*emailOld?* : EMAIL

*f\_name?* : NAME

*m\_name?* : NAME

*l\_name?* : NAME

*image?* : IMAGE

*p\_affiliation?* : NAME

*p\_position?* : POSITION

*p\_address?* : ADDRESS

*p\_url?* : URL

$(emailNew? = emailOld? \vee emailNew? \notin primary\_email) \wedge$   
 $primary\_email' = (primary\_email - \{emailOld?\}) \cup \{emailNew?\} \wedge$   
 $first\_name' = (first\_name - \{emailOld? \rightarrow X\}) \cup \{emailNew? \rightarrow f\_name?\} \wedge$   
 $middle\_name' = middle\_name - \{emailOld? \rightarrow X\} \wedge$   
 $m\_name? \neq \emptyset \Rightarrow middle\_name' = middle\_name \cup \{emailNew? \rightarrow m\_name?\}$   
 $last\_name' = (last\_name - \{emailOld? \rightarrow X\}) \cup \{emailNew? \rightarrow l\_name?\} \wedge$   
 $image' = image - \{emailOld? \rightarrow X\} \wedge$   
 $image? \neq \emptyset \Rightarrow image' = image \cup \{emailNew? \rightarrow image?\}$   
 $primary\_affiliation' = (primary\_affiliation - \{emailOld? \rightarrow X\}) \cup$   
 $\{emailNew? \rightarrow p\_affiliation?\} \wedge$   
 $primary\_position' = (primary\_position - \{emailOld? \rightarrow X\}) \cup$   
 $\{emailNew? \rightarrow p\_position?\} \wedge$   
 $primary\_address' = (primary\_address - \{emailOld? \rightarrow X\}) \cup$   
 $\{emailNew? \rightarrow p\_address?\} \wedge$   
 $primary\_url' = primary\_url - \{emailOld? \rightarrow X\} \wedge$   
 $p\_url? \neq \emptyset \Rightarrow primary\_url' = primary\_url \cup \{emailNew? \rightarrow p\_url?\}$

*DeleteUser*

$\Delta User$

$\Delta Relations$

*email?* : *EMAIL*

---

$\nexists user \rightarrow sko \in Relations.owns \bullet user.primary\_email = email? \wedge$   
 $\nexists user \rightarrow sko \in Relations.published \bullet user.primary\_email = email? \wedge$   
 $primary\_email' = primary\_email - \{email?\} \wedge$   
 $first\_name' = first\_name - \{email? \rightarrow X\} \wedge$   
 $middle\_name' = middle\_name - \{email? \leftrightarrow X\} \wedge$   
 $last\_name' = last\_name - \{email? \rightarrow X\} \wedge$   
 $image' = image - \{email? \leftrightarrow X\} \wedge$   
 $primary\_affiliation' = primary\_affiliation - \{email? \rightarrow X\} \wedge$   
 $primary\_position' = primary\_position - \{email? \rightarrow X\} \wedge$   
 $primary\_address' = primary\_address - \{email? \rightarrow X\} \wedge$   
 $primary\_url' = primary\_url - \{email? \leftrightarrow X\} \wedge$   
 $\forall rel \in Relations.downloaded \bullet rel = user \rightarrow sko \wedge user.primary\_email = email? \wedge$   
 $Relations.downloaded' = Relations.downloaded - \{rel\}$

---

*AddSKO*

$\Delta SKO$

$\Delta Relations$

*uri?* : *URI*

*title?* : *NAME*

*body?* : *BODY*

*content\_type?* : *CTYPE*

*category?* : *STYPE*

*access\_rights?* : *ARIGHTS*

*sko\_visibility?* : *VISIBILITY*

*downloads\_visibility?* : *VISIBILITY*

*comments\_visibility?* : *VISIBILITY*

*reviews\_visibility?* : *VISIBILITY*

*submitted\_to\_review\_visibility?* : *VISIBILITY*

*contributors?* : bag *User*

$\forall x \in \text{contributors?} \bullet \text{receive\_consent}(x, \text{uri?}, \text{contributors?}, \text{access\_rights?}) \wedge$

$\text{uri}' = \text{uri} \cup \{\text{uri?}\}$

$\text{date}' = \text{date} \cup \{\text{uri?} \leftrightarrow \text{current\_system\_date}\}$

$\text{title?} \neq \emptyset \Rightarrow \text{title}' = \text{title} \cup \{\text{uri?} \leftrightarrow \text{title?}\}$

$\text{body?} \neq \emptyset \Rightarrow \text{body}' = \text{body} \cup \{\text{uri?} \leftrightarrow \text{body?}\}$

$\text{content\_type}' = \text{content\_type} \cup \{\text{uri?} \leftrightarrow \text{content\_type?}\}$

$\text{category}' = \text{category} \cup \{\text{uri?} \leftrightarrow \text{category?}\}$

$\text{access\_rights}' = \text{access\_rights} \cup \{\text{uri?} \leftrightarrow \text{access\_rights?}\}$

$\text{sko\_visibility}' = \text{sko\_visibility} \cup \{\text{uri?} \leftrightarrow \text{sko\_visibility?}\}$

$\text{downloads\_visibility}' = \text{downloads\_visibility} \cup \{\text{uri?} \leftrightarrow \text{downloads\_visibility?}\}$

$\text{comments\_visibility}' = \text{comments\_visibility} \cup \{\text{uri?} \leftrightarrow \text{comments\_visibility?}\}$

$\text{reviews\_visibility}' = \text{reviews\_visibility} \cup \{\text{uri?} \leftrightarrow \text{reviews\_visibility?}\}$

$\text{submitted\_to\_review\_visibility}' = \text{submitted\_to\_review\_visibility} \cup$

$\{\text{uri?} \leftrightarrow \text{submitted\_to\_review\_visibility?}\}$

$\forall x \in \text{contributors?} \bullet \exists \text{sko} \bullet \text{sko.uri} = \text{uri?} \wedge$

$\text{Relations.owns}' = \text{Relations.owns} \cup \{x \rightarrow \text{sko}\}$

*CopySKO*

$\Delta SKO$

$\Delta Relations$

*uri1?* : *URI*

*uri2?* : *URI*

*title?* : *NAME*

*body?* : *BODY*

*content\_type?* : *CTYPE*

*category?* : *STYPE*

*access\_rights?* : *ARIGHTS*

*sko\_visibility?* : *VISIBILITY*

*downloads\_visibility?* : *VISIBILITY*

*comments\_visibility?* : *VISIBILITY*

*reviews\_visibility?* : *VISIBILITY*

*submitted\_to\_review\_visibility?* : *VISIBILITY*

$uri' = uri \cup \{uri2?\}$

$date' = date \cup \{uri2? \leftrightarrow current\_system\_date\}$

$title? \neq \emptyset \Rightarrow title' = title \cup \{uri2? \leftrightarrow title?\}$

$body? \neq \emptyset \Rightarrow body' = body \cup \{uri2? \leftrightarrow body?\}$

$content\_type' = content\_type \cup \{uri2? \leftrightarrow content\_type?\}$

$category' = category \cup \{uri2? \leftrightarrow category?\}$

$access\_rights' = access\_rights \cup \{uri2? \leftrightarrow access\_rights?\}$

$sko\_visibility' = sko\_visibility \cup \{uri2? \leftrightarrow sko\_visibility?\}$

$downloads\_visibility' = downloads\_visibility \cup \{uri2? \leftrightarrow downloads\_visibility?\}$

$comments\_visibility' = comments\_visibility \cup \{uri2? \leftrightarrow comments\_visibility?\}$

$reviews\_visibility' = reviews\_visibility \cup \{uri2? \leftrightarrow reviews\_visibility?\}$

$submitted\_to\_review\_visibility' = submitted\_to\_review\_visibility \cup$

$\{uri2? \leftrightarrow submitted\_to\_review\_visibility?\}$

$\exists sko1, sko2 \bullet sko1.uri = uri1? \wedge sko2.uri = uri2? \wedge$

$Relations.version\_of' = version\_of \cup \{sko2 \rightarrow sko1\}$

*DeleteSKO*

$\Delta SKO$

$\Delta Relations$

*uri?* : *URI*

$\nexists user \rightarrow sko \in Relations.published \bullet sko.uri == uri? \wedge$

$\forall X \rightarrow Y \in Relations.on\_field \bullet$

$X.uri == uri? \Rightarrow on\_field' = on\_field - \{X \rightarrow Y\} \wedge$

$\forall X \rightarrow Y \in Relations.Z \bullet$

$Z \in \{owns, downloaded\} \wedge Y.uri == uri?$

$\Rightarrow Z' = Z - \{X \rightarrow Y\} \wedge$

$\forall X \rightarrow Y \in Relations.Z \bullet$

$(Z \in \{review\_of, comment\_on, cites, submitted\_to, extension\_of, variation\_of, version\_of, parent\_of\} \wedge (X.uri == uri? \vee Y.uri == uri?))$

$\Rightarrow Z' = Z - \{X \rightarrow Y\} \wedge$

$uri' = uri - \{uri?\} \wedge$

$date' = date - \{uri?? \rightarrow X\} \wedge$

$title' = title - \{uri?? \leftrightarrow X\} \wedge$

$body' = body - \{uri?? \leftrightarrow X\} \wedge$

$content\_type' = content\_type - \{uri?? \rightarrow X\} \wedge$

$category' = category - \{uri?? \rightarrow X\} \wedge$

$access\_rights' = access\_rights - \{uri?? \rightarrow X\} \wedge$

$sko\_visibility' = sko\_visibility - \{uri?? \rightarrow X\} \wedge$

$downloads\_visibility' = downloads\_visibility - \{uri?? \rightarrow X\} \wedge$

$comments\_visibility' = comments\_visibility - \{uri?? \rightarrow X\} \wedge$

$reviews\_visibility' = reviews\_visibility - \{uri?? \rightarrow X\} \wedge$

$submitted\_to\_review\_visibility' = submitted\_to\_review\_visibility - \{uri?? \rightarrow X\} \wedge$

### ModifyBranchOf

$\Delta$ Relations

$branch\_of\_rel? : OntTerm \rightarrow OntTerm$

$new\_field? : OntTerm$

$branch\_of\_rel? = field1 \rightarrow field2 \wedge$   
 $Relations.branch\_of' = (Relations.branch\_of - \{branch\_of\_rel?\}) \cup$   
 $\{field1 \rightarrow new\_field?\}$

### AddOnField

$\Delta$ Relations

$sco? : SKO$

$field? : OntTerm$

$Relations.on\_field' = Relations.on\_field \cup \{sco? \rightarrow field?\} \wedge$   
 $\forall x \in \{sco2 \mid sco \rightarrow sco2 \in Relations.parent\_of\} \bullet$   
 $x \rightarrow term \notin Relations.on\_field \Rightarrow$   
 $Relations.on\_field' = Relations.on\_field \cup \{x \rightarrow term\}$

### DeletedOnField

$\Delta$ Relations

$on\_field\_rel? : SKO \rightarrow OntTerm$

$on\_field\_rel? = sco \rightarrow ont\_term \wedge$   
 $Relations.on\_field' = Relations.on\_field - \{on\_field\_rel?\} \wedge$   
 $\forall x \in \{sco2 \mid sco \rightarrow sco2 \in Relations.parent\_of\} \bullet$   
 $\left( \exists y \bullet \left( \begin{array}{l} y \rightarrow x \in Relations.parent\_of \wedge \\ y \neq sco \wedge \\ y \rightarrow term \in Relations.on\_field \end{array} \right) \right) \vee$   
 $Relations.on\_field' = Relations.on\_field - \{x \rightarrow term\}$

### AddOwns

$\Delta$ Relations

$sco? : SKO$

$user? : USER$

$access\_rights? : ARIGHTS$

$\forall author \in \{x \mid x == user? \vee (x \rightarrow sco? \in Relations.owns)\} \bullet$   
 $receive\_consent(author, add\_user(user?, sco?, access\_rights?)) \wedge$   
 $Relations.owns' = Relations.owns \cup \{user? \rightarrow sco?\} \wedge$   
 $sco?.access\_rights = access\_rights? \wedge$   
 $\forall x \in \{sco2 \mid sco? \rightarrow sco2 \in Relations.parent\_of\} \bullet$   
 $user? \rightarrow x \notin Relations.owns \Rightarrow$   
 $Relations.owns' = Relations.owns \cup \{user? \rightarrow x\}$

*DeleteOwns*

$\Delta$ Relations

$owns\_rel? : User \rightarrow SKO$

$access\_rights? : ARIGHTS$

$owns\_rel? = user \rightarrow sko \wedge$

$\forall author \in \{x \mid x \rightarrow sko \in Relations.owns\} \bullet$

$receive\_consent(author, delete\_user(user, sko, access\_rights?)) \wedge$

$Relations.owns' = Relations.owns - \{owns\_rel?\} \wedge$

$sko.access\_rights = access\_rights? \wedge$

$\forall x \in \{sko2 \mid sko \rightarrow sko2 \in Relations.parent\_of\} \bullet$

$\left( \exists y \bullet \left( \begin{array}{l} y \rightarrow x \in Relations.parent\_of \wedge \\ y \neq sko \wedge \\ user \rightarrow y \in Relations.owns \end{array} \right) \right) \vee$

$Relations.owns' = Relations.owns - \{user \rightarrow x\}$

*AddPublished*

$\Delta$ Relations

$user? : User$

$sko? : SKO$

$\exists x \bullet x \rightarrow sko? \in Relations.owns \wedge receive\_consent(x, published(user?, sko?)) \wedge$

$Relations.published' = Relations.published \cup \{user? \rightarrow sko?\}$

*DeletePublished*

$\Delta$ Relations

$published\_rel? : User \rightarrow SKO$

$published\_rel? = user \rightarrow sko \wedge$

$Relations.published' = Relations.published - \{published\_rel?\} \wedge$

$\forall x \bullet x \rightarrow sko \in Relations.owns \wedge inform(x, not\_published(published\_rel?)) \wedge$

$inform(user, not\_published(published\_rel?))$

*AddDownloaded*

$\Delta$ Relations

$user? : User$

$sko? : SKO$

$Relations.downloaded' = Relations.downloaded \cup \{user? \rightarrow sko?\}$

*AddParentOf*

$\Delta Relations$

$sko1? : SKO$

$sko2? : SKO$

$(\exists x \rightarrow sko2 \in Relations.owns \bullet receive\_consent(x, parent\_of(sko1?, sko2?)) \vee$

$\exists rel \in Relations.submitted\_to \bullet rel = sko1? \rightarrow sko2?) \wedge$

$Relations.parent\_of' = Relations.parent\_of \cup \{sko1? \rightarrow sko2?\}$

*DeleteParentOf*

$\Delta Relations$

$parent\_of\_rel? : SKO \rightarrow SKO$

$Relations.parent\_of' = Relations.parent\_of - \{parent\_of\_rel?\}$

*AddVariationOf*

$\Delta Relations$

$sko1? : SKO$

$sko2? : SKO$

$Relations.variation\_of' = Relations.variation\_of \cup \{sko1? \rightarrow sko2?\}$

*DeleteVariationOf*

$\Delta Relations$

$variation\_of\_rel? : SKO \rightarrow SKO$

$Relations.variation\_of' = Relations.variation\_of - \{variation\_of\_rel?\}$

*AddExtensionOf*

$\Delta Relations$

$sko1? : SKO$

$sko2? : SKO$

$Relations.extension\_of' = Relations.extension\_of \cup \{sko1? \rightarrow sko2?\}$

*DeleteExtensionOf*

$\Delta Relations$

$extension\_of\_rel? : SKO \rightarrow SKO$

$Relations.extension\_of' = Relations.extension\_of - \{extension\_of\_rel?\}$

*AddSubmittedTo*

$\Delta Relations$

$sko1? : SKO$

$sko2? : SKO$

$\forall author \in \{x \mid x \rightarrow sko1? \in Relations.owns\} \bullet$   
 $receive\_consent(author, submit(sko1?, sko2?)) \wedge$   
 $Relations.submitted\_to' = Relations.submitted\_to \cup \{sko1? \rightarrow sko2?\}$

*DeleteSubmittedTo*

$\Delta Relations$

$submitted\_to\_rel? : SKO \rightarrow SKO$

$\forall author \in \{x \mid x \rightarrow sko1? \in Relations.owns\} \bullet$   
 $receive\_consent(author, cancel\_submit(sko1?, sko2?)) \wedge$   
 $Relations.submitted\_to' = Relations.submitted\_to - \{submitted\_to\_rel?\} \wedge$   
 $\forall author \in \{x \mid x \rightarrow sko2? \in Relations.owns\} \bullet$   
 $inform(author, cancel\_submit(sko1?, sko2?))$

*AddCommentOn*

$\Delta Relations$

$sko1? : SKO$

$sko2? : SKO$

$Relations.comment\_on' = Relations.comment\_on \cup \{sko1? \rightarrow sko2?\}$

*DeleteCommentOn*

$\Delta Relations$

$comment\_on\_rel? : SKO \rightarrow SKO$

$Relations.comment\_on' = Relations.comment\_on - \{comment\_on\_rel?\}$

*AddReviewOf*

$\Delta Relations$

$sko1? : SKO$

$sko2? : SKO$

$\exists x, sko3 \bullet \left( \begin{array}{l} sko2? \rightarrow sko3 \in Relations.submitted\_to \wedge \\ x \rightarrow sko3 \in Relations.owns \wedge \\ x \rightarrow sko2? \in Relations.owns \end{array} \right) \wedge$   
 $Relations.review\_of' = Relations.review\_of \cup \{sko1? \rightarrow sko2?\}$

*DeleteReviewOf*

$\Delta$ *Relations*

*review\_of\_rel?* : *SKO*  $\rightarrow$  *SKO*

*Relations.review\_of'* = *Relations.review\_of* - {*review\_of\_rel?*}

## References

- Ashri, R., Ramchurn, S. D., Sabater, J., Luck, M., and Jennings, N. R. (2005). Trust evaluation through relationship analysis. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*, pages 1005–1011, New York, NY, USA. ACM.
- Heath, T., Motta, E., and Petre, M. (2007). Computing word-of-mouth trust relationships in social networks from semantic web and web2.0 data sources. In *Proceedings of the Workshop on Bridging the Gap between Semantic Web and Web 2.0, 4th European Semantic Web Conference (ESWC '07)*.
- Mui, L. (2003). *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Massachusetts Institute of Technology (MIT).
- Newman, M. E. J. (2001). Who is the best connected scientist? a study of scientific coauthorship networks. *Physical Review E*, 64.
- Pujol, J. M., Sangüesa, R., and Delgado, J. (2002). Extracting reputation in multi agent systems by means of social network topology. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '02)*, pages 467–474, New York, NY, USA. ACM.
- Sabater, J. and Sierra, C. (2002). Reputation and social network analysis in multi-agent systems. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '02)*, pages 475–482, New York, NY, USA. ACM.
- Sierra, C. and Debenham, J. (2008). Information-based reputation. Submitted for publication.

- Sierra, C., Sabater, J., and Osman, N. (2008). Liquidrep. Presented at the Second LiquidPub Project Meeting in Paris, France.
- Spivey, J. M. (1989). *The Z notation: a reference manual*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Yu, B. and Singh, M. P. (2003). Searching social networks. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '03)*, pages 65–72, New York, NY, USA. ACM.