Specifying the Collaborative Tagging System *

Cédric Mesnage
cedric.mesnage@lu.unisi.ch
Faculty of Informatics
University of Lugano
Via G. Buffi 13
6900 Lugano, Switzerland

Mehdi Jazayeri
mehdi.jazayeri@lu.unisi.ch
Faculty of Informatics
University of Lugano
Via G. Buffi 13
6900 Lugano, Switzerland

The world is the totality of facts, not of things.
(Ludwig Wittgenstein, [9], proposition 1.1)

ABSTRACT
The collaborative tagging system is an interaction between humans, terms and objects over time which results in a collectively organized knowledge. In this paper, based on the study of existing software systems, we present the specifications of the collaborative tagging system written in TLA+.

Keywords
Collaborative Tagging, Specifications, TLA+, precise description

1. INTRODUCTION
To analyse emergent web applications scientifically, we need precise abstract definitions of the fundamental elements interacting in the systems in which these applications are successful. In this paper, we present the logic specifications of the abstractions interacting in the collaborative tagging system.

Long-living software systems grow more complex over time. The behavior of such systems is not always well-understood and is often a source of maintenance and extendibility problems. The Internet, which started as a simple protocol (IP), is now one of the more complex software systems ever assembled, consisting of millions of components and applications interacting with millions of users. New applications on the Internet emerge regularly introducing new concepts and behaviors. These new concepts interact in sometimes unintended ways with other parts of the Internet making it ever more difficult to understand the behavior of the Internet. It is clearly not possible to formally specify the workings of the entire Internet. However, to be able to understand, explain, and extend the behavior of aspects of any complex system, we need a high-level precise description of those aspects. For a system as large and complex as the Internet, we should be able to build such a high-level description incrementally and independently for different aspects. In this paper, we show how this can be done for collaborative tagging, one of the emerging popular applications on the Internet.

Collaborative tagging [7] is an extension of the tagging activity introduced in several popular websites such as Flickr or del.icio.us. It has proven successful in these websites as a way of sharing resources (i.e. photos, bookmarks).

Collaborative tagging has also attracted the interest of researchers [7] who are trying to understand the foundations of the activity, its current implementations, and possible role in the collective intelligence idea. Collaborative tagging is not merely a computer science problem in that it attempts to enable the collaboration of large numbers of humans in collectively reviewing and structuring large amounts of information with little restrictions imposed by the system. Many contributions have been done lately[10, 6, 8, 1, 4]. We refer primarily to [1] and [4].

This paper has two primary goals: to show a precise description (specification) of collaborative tagging, and to give an example of incremental derivation of specifications for Internet-based services.

1.1 Definitions
Since this paper is about precise descriptions, we start with a set of definitions in order to avoid misunderstanding on the meaning of the main terms we use. Many of these definitions can be found in the Oxford Dictionary. The glossary is given in Figure 1.

To avoid ambiguity inherent in the use of the word "tag" as both noun and verb, in this paper we use the word "tag" only as a verb, that is, the actual act of tagging. For noun usage, we use "term" to name the keyword used to tag. This avoids confusion and enhances clearness of the specifications.

The collaborative tagging system is an interaction between humans, terms and objects over time which results in a collectively organized knowledge. In this paper we present and explain specifications of this system. The next subsections of

*This work is funded by the European Project "Nepomuk, the social semantic desktop". http://www.semanticdesktop.org

Specification In [2] Michael Jackson claims that

> The terminology of software development is mostly in a chaos that correctly reflects the chaotic state of the field. Usage of the word specification is no exception.
> (Michael Jackson, page 193)

But here we take as definition

> A specification is a written description of what a system is supposed to do.
> (Leslie Lamport, [3], paragraph 4, page 1)

Collaborative produced or conducted by two or more parties working together.

Tagging attach a label to.

System a set of connected things or parts forming a complex whole.

Engineering Engineering requires not just the ability to use mathematics, but the ability to understand what, if anything, the mathematics tells us about an actual system. (Leslie Lamport, [3], page 21)

Behavior The way in which a natural phenomenon or a machine works or functions.

Abstraction The process of considering something independently of its associations, attributes, or concrete accompaniments.

Trust Firm belief in the reliability, truth, ability, or strength of someone or something.

Term A word or phrase used to describe a thing or to express a concept, esp. in a particular kind of language or branch of study.

Human A human being, esp. a person as distinguished from an animal or (in science fiction) an alien.

Object A thing external to the thinking mind or subject.

Phenomenon The object of a person's perception; what the senses or the mind notice.

Observation The action or process of observing something or someone carefully or in order to gain information.

Representation A mental state or concept regarded as corresponding to a thing perceived.

Transcription A written or printed representation of something.

Memory Something remembered from the past.

Figure 1: Fundamental glossary.

this introduction describe briefly and informally this system as seen in existing software and present the methodology we choose to write the specifications. In order to facilitate the understanding of the specifications, we describe the abstractions involved in the collaborative tagging system specification and first describe the collaborative tagging memory as a module. We use model-checking to demonstrate the validity of these specifications. Finally, we propose how our work can be extended in different directions.

1.2 The Collaborative Tagging System

The collaborative tagging system emerges from many new software systems (i.e Del.icio.us, Flickr, Technorati, Connotea, CiteUlike, RawSugar...). It is a way of categorizing knowledge by freely assigning sets of terms to objects (often to web pages or bookmarks).

A fundamentally new concept introduced by the new generation of Web applications such as collaborative tagging is to include and *trust* the user in further evolution of the system (and its associated information). In fact, anyone can create new terms in a simple way. The users feel that trust, and we believe it explains the success of these software systems.

Another important aspect is collaboration. In traditional software, an expert or a set of experts creates a thesaurus of categories and takes care of indexing documents themselves. This avoids the insertion of erroneous data, but also the insertion of different aspects and viewpoints on the objects. The collaborative tagging system allows anyone to index any object without any restriction with meaningful terms. The notion of meaningfulness is subjective, but the combination of points of views from many people is more complete than the one of only a chosen subset (experts). The resulting taxonomy is often called a *folksonomy*.

Though simple in terms of software, this system exhibits complex collective behaviors. As the number of people working collectively grows, the system of interaction between them must be simplified. In a democratic political system, the voting system is an example of a simple interaction to decide collectively at a large scale. To organize knowledge together, the interaction system between humans must be as simple as possible.

Recent fundamental work as been done by analysing Del.icio.us[1] and Flickr[4] tagging memory. In [1], Golder compares the tagging system to a typical filesystem organization

> In contrast to a hierarchical file system, a non-exclusive, flat tagging system could, unlike the system described above, identify such an article as being about a great variety of things simultaneously...

They describe how *synonymous* and *polysemous* aspects of terms can be solved in a collaborative tagging system and define tagging as

> Tagging is fundamentally about sensemaking.

They identified different ways people use terms to tag

- Identifying what (or who) it is about.
- Identifying what it is.
- Identifying who owns it.
- Refining categories.
- Identifying qualities or characteristics.
- Self reference.
- Task organizing.

in [4] Marlow gives a model of tagging which is represented as a bipartite graph showing resources (what we call objects in this paper) on the left hand side and users on the right hand side. The edges between resources and users carry some terms and represent an act of tagging (what we call an observation in the following sections). He also identifies some aspects of tagging software systems :

- Tagging Rights.
- Tagging Support.
- Aggregation.
- Type of Object.
- Source of Material.
- Resource Connectivity.
- Social Connectivity.

These empirical studies, together with our own understanding of existing software systems have been helpful in writing the collaborative tagging specifications. In addition to these two works, there has been considerable research activity in this area recently [7].

1.3 Writing Specifications in TLA+

The role of specifications is both well-known and controversial in the practice of software development. Here we are dealing with specifications of a system, as opposed to a specification of a design or architecture. We are using the term and the process in the sense of Lamport:

> Specifying a system helps us to understand it.
>
> (Leslie Lamport, [3], paragraph 4, page 1)

The specification then can be used as a medium of communication among people about the system. The specification may, for example, be used by software architects to design a particular implementation of the system. The specifications are thus a bridge between the requirements and the implementation. They are also a means of defining the relationship between the system and the real world. Thus, specifications must be specified at a much higher level of abstraction than implementation-oriented notations. Typically, a logical notation is most appropriate. For this purpose, we have chosen to use TLA+ (Temporal Logic of Actions). We do not discuss here the different values of such languages and choose to use TLA+. Any number of equivalent notations could be used for this purpose. One advantage of TLA is that it comes with a model checker (TLC), which efficiently validates the model and its properties.

Temporal Logic of Actions is a first order logic enriched with temporal operators. We describe in this section the key aspects of such specifications and the language specifics. We emphasize that the main purpose of writing specifications is to understand the system. According to Lamport:

> The hardest part of writing a specification is choosing the proper abstraction.
>
> The art of abstraction is learned only through experience.
>
> (Leslie Lamport, [3], page 24)

Thus a large part of our effort has gone into deciding the abstractions to specify. We hope to give the reader the benefit of our experience gained in formulating the specifications. Note that such specifications may be defined for a system that is yet to be built (forward engineering) in order to understand the requirements and help in evaluating the design alternatives, or for an existing system in order to understand it better and help in enhancing it (reverse engineering).

In our case, the purpose of the specifications is to formally define possible behaviors of the studied system. A behavior is described by Leslie Lamport as

> A behavior is an infinite sequence of states.
>
> A behavior describes a potential history of the universe.
>
> (Leslie Lamport, [3], paragraph 4, page 18)

but for practical reasons, we consider only a subset of the abstractions which make sense to consider in the study of a particular system. In Figure 2, we present a behavior of the *HourClock* example shown in [3]. In this representation only two *variables* are considered, *hr* the current hour and *tmp* the current temperature. Each *state* is represented by the values of the variables. The process of writing specifications is to first identify this kind of behavior and then formally define the *actions* which describe a *correct behavior*.

A specification in TLA+ (despite the declaration of variables and extensions needed) starts with a predicate which describes the potential initial state of a behavior. This is usually a conjunction of expressions describing the possible values of the considered variables. The following predicate describes the initial state of the temperature hour clock example. The $\triangleq$ is read "is defined as equal to".

To describe the evolution from one state to another, one must define *actions* which are formulas containing variables

$$\left[\begin{array}{lcr} hr & = & 11 \\ tmp & = & 23.5 \end{array}\right] \rightarrow \left[\begin{array}{lcr} hr & = & 12 \\ tmp & = & 23.5 \end{array}\right] \rightarrow$$
$$\left[\begin{array}{lcr} hr & = & 12 \\ tmp & = & 23.4 \end{array}\right] \rightarrow \left[\begin{array}{lcr} hr & = & 12 \\ tmp & = & 23.3 \end{array}\right] \rightarrow$$
$$\left[\begin{array}{lcr} hr & = & 1 \\ tmp & = & 23.3 \end{array}\right] \rightarrow \cdots$$

Figure 2: An example behavior of the Hour Clock, which also takes care of the temperature.

$$\begin{array}{l} HCTMPInit \triangleq \\ \quad \wedge\, hr \in \{1,\ \ldots,\ 12\} \\ \quad \wedge\, tmp \in Reals \end{array}$$

enriched with a $'$ which represent the value of the variable in the next state. The following predicate defines the hour clock main action (which is to go from one hour to another and to pass to the value 1 after the 12), we don't consider the temperature here:

$$\begin{array}{l} HCini \triangleq hr \in \{1,\ \ldots,\ 12\} \\ HCnxt \triangleq hr' = (hr\%12) + 1 \end{array}$$

A specification is a single formula which should satisfy both the *initial predicate* and the *next-state actions* for any step of the possible behaviors of the system. That is where we use the temporal operator $\Box$ called "box" which expresses a formula which is "always true". We can then write the specifications of the hour clock as follows

$$HCSpec \triangleq HCini \wedge \Box HCnxt$$

But as we have seen in Figure 2, some steps do not change the value of the variable *hr* especially if we consider a state being a state of the whole universe. Temporal logic of actions here introduce the concept of *stuttering steps* which consider the possibility of having one or more variables left unchanged from one state to another If we consider the variable *hr* as the stuttering variable, we write it as follows:

$$HCSpec \triangleq HCini \wedge \Box[HCnxt]_{hr}$$

We now have the *HCSpec* formula which, if satisfied by a behavior, validates the behavior of the described system. This short description of the TLA+ language should be enough to enable the reader to understand the following system specifications. The language is described fully in [3] which is also available on the TLA Web page[1]. The TLA system is accompanied by a model checker, TLC, that may be used to validate properties of well-defined specifications.

2. THE COLLABORATIVE TAGGING SYSTEM SPECIFICATIONS

To explain the specifications we first need to go back to the abstractions of the collaborative tagging system. The collaborative tagging system is an interaction between humans, terms and objects. In Figure 3 we give a graphic representation of this interaction which we explain in this section.

The environment of the collaborative tagging system is the set of all *phenomena* emerging from the real world. The phenomena are observable facts. Differing from typical semantic systems, the facts are considered outside of the system. Only the human perceptions of the facts are considered valuable. We do not have direct access to the facts, neither can we formulate them. We can feel an object, but we do not know what it really is.

Through the senses, humans perceive the phenomena. The resulting *observations* are the data of our system. Indeed the only thing that is known are the observations. Observations are subjective, therefore different humans will have different observations of the same phenomenon. Some perceive the color, some others the shape or any other aspect. A fundamental premise of collaborative tagging is that the combination of all these observations is the best picture we can get of the real phenomenon which occurred.

When a human observes a phenomenon, he thinks a mental *representation*. This helps him to reason about the phenomenon. Depending on the aspects he perceived in his observation, he mentally associates terms to the phenomenon. These terms may be different for each human who observed the phenomenon so we need to consider them all.

To be able to combine the mental representations of different humans, we first need humans to express them. They can not directly express their representations so they share observations together with the terms they use in their mental scheme. This we call *transcription*. In the collaborative tagging system, humans write their observations by tagging.

When one assigns a tag to an object, the system stores the observation in a *memory*. This memory is the set of all transcripted observations. A written observation is the association of the human (the writer), the object of the observation, the term of his representation and the date when he wrote it.

Other humans can then *retrieve* these observations from the *collaborative tagging memory*. When reading the observations of other humans, the reader changes his representations of the object as he now perceives the observations made by different humans.

Reading and writing are the two operations of the memory, tagging and retrieving the two operations of the collaborative tagging system. We present in the next subsections the specification of the memory and of the whole system.

The abstractions and the specifications presented here reflect our understanding of many existing software systems and of an ideal one. We try to be as generic and general as possible to specify *the* collaborative tagging system and not *a* collaborative tagging software system.

[1]http://research.microsoft.com/users/lamport/tla/tla.html

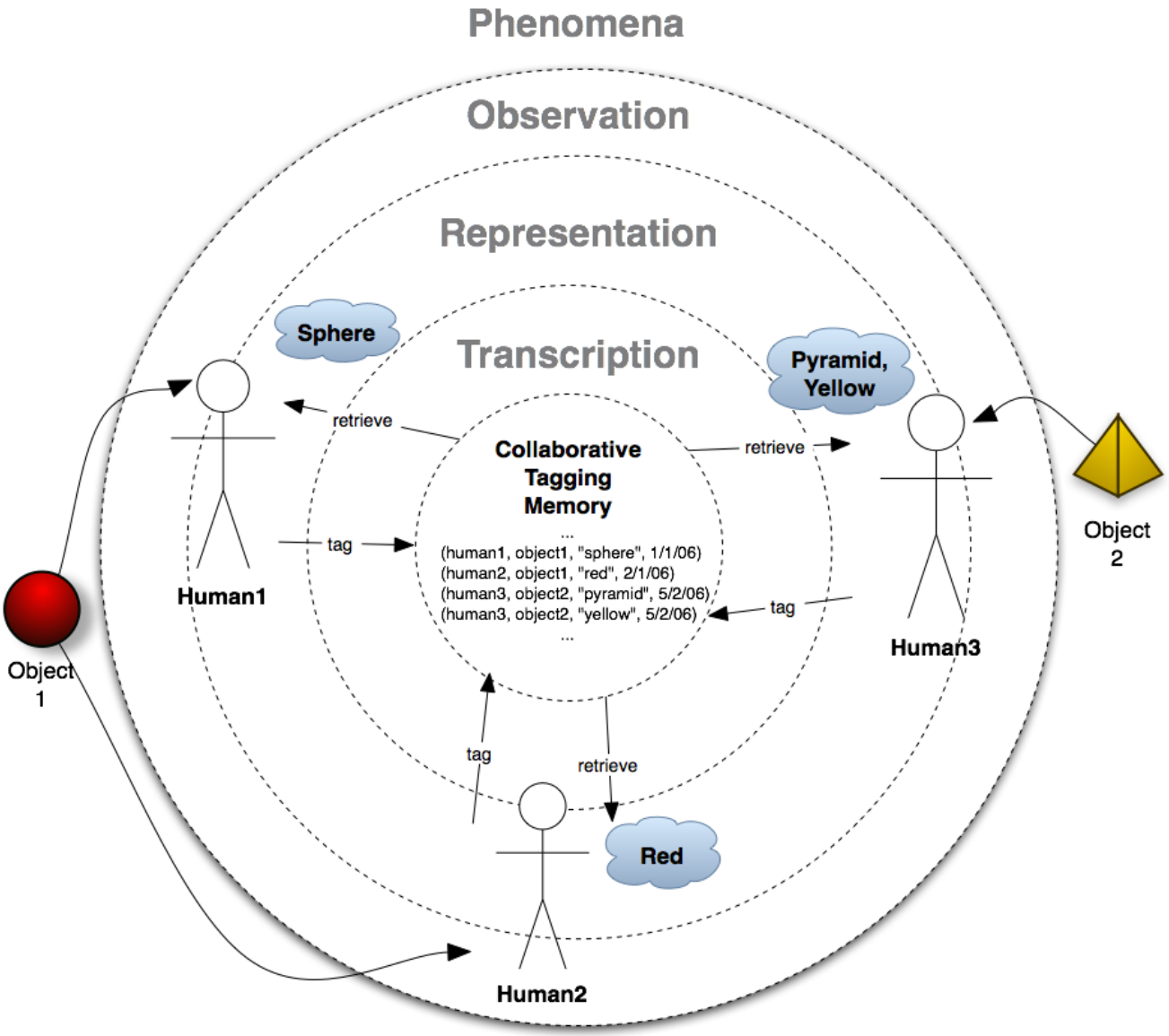

Figure 3: Diagram of the collaborative tagging system and its environment.

The phenomena are the environment of the system. Observations are the human perceptions of the phenomenon. The representations are the mental association of terms to the observations. The transcription is the process of writing the observations, what we call tag. The collaborative tagging memory stores observations.

2.1 The Collaborative Tagging Memory

We define a module to specify the collaborative tagging memory. The full specification of this module is presented in Figure 5. The predicate on line 4 defines the initial state of this system. We need to consider the time, represented by the variable *now* which is initially any value in the *Time* set(line 5). The *Observations* is the variable which stores the data of our memory(line 6). In any initial state, the observations set may contain any observation that has been done in the past. One particular observation is defined as a record which consists of values for a human, an object, a term and a date. Records allow us to write expressions like *observation.human* which represents the value of the human in this particular observation. The set observation in the initial state is a subset of the set of all records for any human, object, term and date of the past(line 7-8). The variable *buffer* is a function which for any human returns a set of observations he requested. In the initial state this is set to the empty set for any human as no one has requested anything yet(line 9).

From the definition of the initial predicate, we can determine the constants and variables we use. The constants can be seen as parameters of the system. When using the TLC model checker, the constants are set to particular values. On line 3 we define the constants *Humans*(the set of all humans interacting with the memory), *Terms*(the set of all terms used in the memory), *Objects*(the set of all objects observed), *Time*(the set of all values of time considered). This is a tradeoff as new humans might enter the system during its lifetime, new terms could be created as new objects defined, and the time passes without stopping. In a realistic system, these constants would be variables. The reader can understand constants as types.

The variables defined on line 4 represent what change during the life of the system. In this case the behavior is represented by the value of the *Observations*(the set of all observations written so far), *now* the time of now in number of seconds, *buffer*(the state of the requests made by readers).

The type invariant property on line 16 defines that the behavior must respect the *TypeInvariant* for every state. The type invariant (line 10-15) is quite straightforward as the variables *now* and *Observations* must be in any state allowed by the initial predicate. The function buffer must respect that for any reader, the set of observations requested is a subset of the stored observations(line 15).

This system being a memory storing the result of transcriptions, its fundamental operations are defined as *read*(line 19) and *write*(line 17). The *write* operator defines the next state of the variable *Observations* as being the union of the current value of it and of the new observation that is written.

The *read* operator is a bit more complex as it takes a *reader* and a set of *requests*. It defines the value of the next state of the function *buffer* and changes the value associated to the human *reader*. It sets this to the set of all observations which satisfy at least one request(line 20-22). Each request is a record consisting of a type(either human, object, term or date) and a value that is requested for that type. The reader can then feel free to read the buffer prepared for him.

The next state actions of the memory are defined as *Write*(line 23), *Read*(line 27) and *NextNow*(line 31). A *Write* action occurs if there exists a human, an object and a term such that the operation *write* for the time of now is true(line 24-25). A write action leaves the variables *buffer* and *now* unchanged(line 26).

The *Read* action occurs if there exists a human, an object or a term such that the operation read is true for the human as the reader and the requests on the term and the object(line 28-29). The read operation changes the *buffer* but leaves the variables *now* and *Observations* unchanged(line 30).

The *NextNow* action adds a second to the *now* variable if the resulting value is still in Time. It leaves the variables *buffer* and *Observations* unchanged(line 32).

We now define the next state action of the whole memory *NextCollaborativeTaggingMemory* as being the disjunction of the actions *Write*, *Read* and *NextNow*(line 33-34).

The specification of the memory is defined by the predicate *CollaborativeTaggingMemorySpec* which is the conjunction of the initial predicate and the temporal expression which reflects that for all states, the next state action is true. There can be stuttering steps which leave the value of the variables unchanged(line 37).

We now have the full specification of the collaborative tagging memory. In the following subsection we show how we used the model checker to verify it.

Model Checking The Collaborative Tagging Memory Module

The TLA+ language comes with a model checker, TLC. We used it all along the writing process and it has been valuable in the refinement of the specifications. The model checker takes a configuration file which give values to the constants and states the properties to be checked. Figure 4 shows the configuration file we use to model check the collaborative tagging memory.

```
CONSTANTS
    Humans = {"human:bob", "human:alice"}
    Objects = {"object:1","object:2"}
    Terms = {"term:1","term:2"}
    Time = {1,2}
SPECIFICATION CollaborativeTaggingMemorySpec
PROPERTY TypeInvariantProperty
```

Figure 4: The configuration file used to model check the collaborative tagging memory specification.

The model checker generated 66307 states and found 65792 distincts states without errors. The difference reflects the number of ways which reach states with same values. It means that the type invariant property is respected for any state and that the specification permits to cover all these states.

```
MODULE CollaborativeTaggingMemory
EXTENDS Naturals, TLC
CONSTANTS Humans, Terms, Objects, Time
VARIABLES Observations, now, buffer

InitCollaborativeTaggingMemory ≜
  ∧ now ∈ Time
  ∧ Observations ∈
    SUBSET {[human ↦ human, object ↦ object, term ↦ term, date ↦ date] :
      human ∈ Humans, object ∈ Objects, term ∈ Terms, date ∈ {t ∈ Time : t ≤ now}}
  ∧ buffer = [human ∈ Humans ↦ {}]

TypeInvariant ≜
  ∧ now ∈ Time
  ∧ Observations ∈
    SUBSET {[human ↦ human, object ↦ object, term ↦ term, date ↦ date] :
      human ∈ Humans, object ∈ Objects, term ∈ Terms, date ∈ {t ∈ Time : t ≤ now}}
  ∧ ∀ reader ∈ Humans : buffer[reader] ∈ SUBSET Observations

TypeInvariantProperty ≜ □ TypeInvariant

write(writer, object, term, date) ≜   Writing adds a new observation.
  Observations′ = Observations ∪ {[human ↦ writer, object ↦ object, term ↦ term, date ↦ date]}

read(reader, requests) ≜   Reading puts the requested observations in a buffer for the reader.
  buffer′ = [buffer EXCEPT ![reader] = {
    observation ∈ Observations : ∃ request ∈ requests :
      observation[request.type] = request.value}]

Write ≜
  ∃ human ∈ Humans : ∃ term ∈ Terms : ∃ object ∈ Objects :
    ∧ write(human, object, term, now)
    ∧ UNCHANGED ⟨now, buffer⟩

Read ≜
  ∃ human ∈ Humans : ∃ term ∈ Terms : ∃ object ∈ Objects :
    read(human, {[type ↦ "term", value ↦ term], [type ↦ "object", value ↦ object]})
    ∧ UNCHANGED ⟨now, Observations⟩

NextNow ≜
  (now + 1) ∈ Time ∧ now′ = now + 1 ∧ UNCHANGED ⟨Observations, buffer⟩

NextCollaborativeTaggingMemory ≜   A next state is defined by either reading, writing or passing the time.
  Read ∨ Write ∨ NextNow

CollaborativeTaggingMemorySpec ≜
  ∧ InitCollaborativeTaggingMemory
  ∧ □[NextCollaborativeTaggingMemory]⟨now, Observations, buffer⟩
```

Figure 5: The CollaborativeTaggingMemory specification

2.2 The Collaborative Tagging System

Figure 7 presents the full collaborative tagging system specifications. According to Figure 3, the collaborative tagging system contains a collaborative tagging memory that we specified in the previous subsection. We instanciate the memory module (line 4) and define as constants and variables the same one of this module. We introduce a new variable *Representation* which represents the state of the mind of each human. The initialization predicate consists of the initialization of the memory and the definition of the representations variable(line 11-12). A representation is a record which consists of a human, an object and a term; initially, the representations is any subset of all possible representations. We explain line 13 at the end of this subsection.

The two actions shown in Figure 3 are *tag*(line 21) and *retrieve*(line 19). These are both interfaces to the collaborative tagging memory.

The next state actions are defined as *Tag* and *Retrieve* predicates on lines 30 and 23. The *Retrieve* action occurs if the reader has a representation which can help him to read observations from the memory which change his representation (lines 25-29). The *Tag* action occurs if the writer has a representation that he can write as an observation(lines 31-33).

The next state action *Next* is defined as the disjunction of the *NextNow* action from the memory and for any human either the *Tag* or the *Retrieve* action(line 34-36).

The full specification predicate is shown on lines 37 and 38. It is the conjunction of the *Init* predicate and for all states of the *Next* action.

We introduce here a theorem that is implied by the specification: "Humans know something about the world". This is defined on lines 5 to 8 as being equals to: for all humans there exists a representation owned by the human. Everyone knows at least something about the world, otherwise he could not tag, neither retrieve anything and the property *Consistency* would never occur. The consistency property is defined as eventually every human has the same representation(line 14-18).

Model Checking The Collaborative Tagging Specification

To model check the theorem presented previously, we need to check that the property *ConsistencyProperty* is respected. We introduce in the *Init* conjunction the predicate *HumansKnowSomethingAboutTheWorld*. Otherwise the property never occurs. This is called an implied init. Figure 6 presents the configuration file used to model check the collaborative tagging system.

The data given as argument to the model checking is the same we used to model check the collaborative tagging memory. TLC found 65792 distinct states and 329216 states were generated. The number of distincts states is the same as with the collaborative tagging memory, which sounds natural as we use the same data and that the tag and retrieve actions are mappings to the read and write actions. This gives us confidence in the validity of both specifications as we obtain the same results.

```
CONSTANTS
    Humans = {"human:bob", "human:alice"}
    Objects = {"object:1","object:2"}
    Terms = {"term:1","term:2"}
    Time = {1,2}

SPECIFICATION CollaborativeTaggingSpec
PROPERTIES ConsistencyProperty TypeInvariantProperty
```

Figure 6: The configuration file used to model check the collaborative tagging system specification.

3. FUTURE WORK

3.1 Possible next steps

Given a valid specification of collaborative tagging as presented here, there are several ways that it can be exploited. The most concrete step is to use it as a guide in design and architecture of an implementation. For example, the specification can be used to guide the choice of data structures and representations for observations. We discuss this in the next subsection.

Another possibility is to explore enhancements to collaborative tagging in a semantic way. For example, what is the effect of associating several terms at the same time with an object? Is that different from assigning each of the terms separately?

Third, we could define other Internet services such as collaborative search and then explore the interaction of these services with collaborative tagging. This, indeed, can be the way to gain control over the ever-expanding set of services and understand their implications as they are introduced.

3.2 Implementation proposals

We are currently working on the implementation of these specifications in the *Gnowsis* semantic desktop [5], defined as:

"A Semantic Desktop is a device in which an individual stores all her digital information like documents, multimedia and messages. These are interpreted as Semantic Web resources, each is identified by a Uniform Resource Identifier (URI) and all data is accessible and queryable as RDF graph. Resources from the web can be stored and authored content can be shared with others. Ontologies allow the user to express personal mental models and form the semantic glue interconnecting information and systems. Applications respect this and store, read and communicate via ontologies and Semantic Web protocols. The Semantic Desktop is an enlarged supplement to the user memory."

In the specifications, we did not care if the tagging memory is distributed or centralized, for implementation purposes in the case of a desktop software system, we propose to implement this memory as distributed amongst users and shared through the use of RSS (Really Simple Syndication) using the XML format of Figure 8.

```
MODULE CollaborativeTagging
EXTENDS Naturals, TLC
CONSTANTS Humans, Terms, Objects, Time
VARIABLES Observations, Representations, buffer, now
INSTANCE CollaborativeTaggingMemory

HumansKnowSomethingAboutTheWorld ≜ Everyone has at least one representation of something.
  ∀ human ∈ Humans :
    ∃ representation ∈ Representations :
      representation.human = human

Init ≜ The memory is initialized, every human has representations.
  ∧ InitCollaborativeTaggingMemory
  ∧ Representations ∈ SUBSET {[human ↦ human, object ↦ object, term ↦ term] :
    human ∈ Humans, object ∈ Objects, term ∈ Terms}
  ∧ HumansKnowSomethingAboutTheWorld

Consistency ≜ Eventually everyone has the same representation of every object.
  ∀ human ∈ Humans : ∀ other ∈ (Humans \ {human}) :
    {⟨r.object, r.term⟩ : r ∈ {r ∈ Representations : r.human = human}} =
    {⟨r.object, r.term⟩ : r ∈ {r ∈ Representations : r.human = other}}

ConsistencyProperty ≜ ◇Consistency

retrieve(reader, requests) ≜ The retrieve action is defined as an interface to reading from the memory.
  read(reader, requests)

tag(writer, object, term, date) ≜ The tag action is defined as an interface to writing to the memory.
  write(writer, object, term, date)

Retrieve(reader) ≜ When someone retrieves observations, it changes his representation.
  UNCHANGED ⟨now, buffer, Observations, Representations⟩ ∧
  ∀ representation ∈ Representations :
    ∧ representation.human = reader
    ∧ retrieve(reader, {[type ↦ "object", value ↦ representation.object], [type ↦ "term", value ↦ representation.term]})
    ∧ Representations′ = Representations ∪
      {[human ↦ reader, object ↦ observation.object, term ↦ observation.term] : observation ∈ buffer[reader]}

Tag(writer) ≜ When someone has representations, he can write them.
  UNCHANGED ⟨Representations, buffer, now, Observations⟩ ∧
  ∀ representation ∈ Representations :
    representation.human = writer ∧ tag(writer, representation.object, representation.term, now)

Next ≜ The time passes or humans can either tag or retrieve.
  ∨ (NextNow ∧ UNCHANGED ⟨Representations, buffer, Observations⟩)
  ∨ (∀ human ∈ Humans : Tag(human) ∨ Retrieve(human))

CollaborativeTaggingSpec ≜
  Init ∧ □[Next]⟨now, Observations, Representations, buffer⟩

THEOREM CollaborativeTaggingSpec ⇒ HumansKnowSomethingAboutTheWorld ∧ ConsistencyProperty
```

Figure 7: The CollaborativeTagging system specification

The specifications of the collaborative tagging system helped us to understand the least data needed in order for a software system to behave like the collaborative tagging system. In these specifications, an observation is a relation between a human, an object, a term and a date. If we consider the human and the object being identified by URIs (Uniform Resource Identifier) as they are all considered as resources in current systems, the term as a string, and the date as a date, we can determine the collaborative tagging XML schema which defines *CollaborativeTaggingMemory* consisting of one or more *observations* as follows. (This schema will be documented and available soon on the collaborative-tagging.org website.) :

```
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace=
  "http://collaborativetagging.org/schema"
 xmlns:xct="http://collaborativetagging.org/schema">
 <xs:complexType name="observation">
  <xs:attribute name="human" type="xs:anyURI"/>
  <xs:attribute name="object" type="xs:anyURI"/>
  <xs:attribute name="term" type="xs:string"/>
  <xs:attribute name="date" type="xs:date"/>
 </xs:complexType>

 <xs:complexType name="CollaborativeTaggingMemory">
  <xs:element type="observation"/>
 </xs:complexType>
</xs:schema>
```

Figure 8: The XML Schema of the collaborative tagging observations.

The specifications lead to an understanding of the services provided by both the collaborative tagging system and the collaborative tagging memory. This simplifies the writing of WSDL(Web Services Description Language) documents describing these services. We plan to do it lately together with the implementation of these services within *Gnowsis*.

4. CONCLUSION

This paper contributes to the emerging collaborative tagging field with the formal specifications in TLA+ of both the collaborative tagging memory and the collaborative tagging system. We gave confidence in the validity of these by using the TLC model checker. It results in a better understanding of possible software systems respecting these specifications. We enunciated a theorem resulting from this understanding. We proposed an XML schema to uniformly represent the collaborative tagging observations.

We hope that others will enhance and continue to validate the properties of the collaborative tagging system specification. Over time, modular and incremental specifications, such as given here, are the only way to understand and explain how complex, interacting, evolving systems work. We hope that a collection of such specifications about important services on the Internet will enable us to reason about the interactions of existing and emerging services and their behaviors.

5. REFERENCES

[1] Scott Golder and Bernardo A. Huberman. The structure of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, April 2006.

[2] Michael Jackson. *Software requirements & specifications: a lexicon of practice, principles and prejudices*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.

[3] Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[4] Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. Position Paper, Tagging, Taxonomy, Flickr, Article, ToRead. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, May 2006.

[5] Leo Sauermann, Ansgar Bernardi, and Andreas Dengel. Overview and outlook on the semantic desktop. In *Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference*, 2005.

[6] Patrick Schmitz. Inducing ontology from flickr tags. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, May 2006.

[7] Frank Smadja, Andrew Tomkins, and Scott Golder. Collaborative web tagging workshop. In *WWW2006, Edinburgh, Scotland*, 2006.

[8] Jennifer Trant and Bruce Wyman. Investigating social tagging and folksonomy in art museums with steve.museum. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, May 2006.

[9] Ludwig Wittgenstein. *Tractatus Logico-Philosophicus*. Routledge, 1922.

[10] Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the semantic web: Collaborative tag suggestions. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, May 2006.