ADVANCES IN INTELLIGENT SYSTEMS AND COMPUTING 172

Lorna Uden
Francisco Herrera
Javier Bajo Pérez
Juan Manuel Corchado Rodríguez (Eds.)

7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing

Springer

Advances in Intelligent Systems and Computing 172

Editor-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

For further volumes:
http://www.springer.com/series/11156

Lorna Uden, Francisco Herrera, Javier Bajo Pérez, and Juan Manuel Corchado Rodríguez (Eds.)

7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing

Springer

Editors

Lorna Uden
FCET
Staffordshire University
The Octagon, Beaconside
UK

Javier Bajo Pérez
Faculty of Computer Science
Pontificial University of Salamanca
Salamanca
Spain

Francisco Herrera
Department Computer Science and
Artificial Intelligence
University of Granada
Granada
Spain

Juan Manuel Corchado Rodríguez
Department of Computing Science
and Control
Faculty of Science
University of Salamanca
Salamanca
Spain

ISSN 2194-5357
e-ISSN 2194-5365
ISBN 978-3-642-30866-6
e-ISBN 978-3-642-30867-3
DOI 10.1007/978-3-642-30867-3
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012939635

© Springer-Verlag Berlin Heidelberg 2013
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Knowledge is increasingly recognised as the most important resource in organisations and a key differentiating factor in business today. It is increasingly being acknowledged that Knowledge Management (KM) can bring about the much needed innovation and improved business performance in organisations. The service sector now dominates the economies of the developed world. Service innovation is fast becoming the key driver of socio-economic, academic and commercial research attention. Knowledge Management plays a crucial role in the development of sustainable competitive advantage through innovation in services. There is tremendous opportunity to realise business value from service innovation by using the knowledge about services to develop and deliver new information services and business services.

Although there are several perspectives on KM, they all share the same core components, namely: People, Processes and Technology. Organisations of all sizes across nearly every industry are seeking new ways to address their knowledge management requirements. Cloud computing offers many solutions to the problems facing KM implementation. Cloud computing is an emerging technology that can provide users with all kinds of scalable services, such as channels, tools, applications, social support for users' personal knowledge amplification, personal knowledge use/reuse, and personal knowledge sharing.

The seventh International Conference on Knowledge Management in Organizations (KMO) offers researchers and developers from industry and the academic world to report on the latest scientific and technical advances on knowledge management in organisations. It provides an international forum for authors to present and discuss research focused on the role of knowledge management for innovative services in industries, to shed light on recent advances in cloud computing for KM as well as to identify future directions for researching the role of knowledge management in service innovation and how cloud computing can be used to address many of the issues currently facing KM in academia and industrial sectors. This conference provides papers that offer provocative, insightful, and novel ways of developing innovative systems through a better understanding of the role that knowledge management plays.

The KMO 2012 proceedings consist of 53 papers covering different aspects of knowledge management and service. Papers came from many different countries including Australia, Austria, Brazil, China, Chile, Colombia, Denmark, Finland, France, Gambia, India, Japan, Jordan, Netherlands, Malaysia, Malta, Mexico, Netherlands, New Zeland, Saudi Arabia, Spain, Slovakia, Slovenia, Taiwan, Turkey, United States of America and United Kingdom. We would like to thank our program committee, reviewers and authors for their contributions. Without their efforts, there would be no conference and proceedings.

Salamanca
July 2012

Lorna Uden
Francisco Herrera
Javier Bajo Pérez
Juan Manuel Corchado Rodríguez

Organization

Conference Chair

Dr. Lorna Uden	Staffordshire University, UK

Program Chair

Dr. Francisco Herrera	University of Granada, Spain

Program Committee

Dr. Senen Barro	University of Santiago de Compostela, Spain
Dr. Hilary Berger	UWIC. England. UK
Dr. Pere Botella	Polytechnic University of Catalonia – BarcelonaTech, Spain
Dr. Vicente Botti	Polytechnic University of Valencia, Spain
Dr. Senoo Dai	Tokyo Institute of Technology, Japan
Dr. Flavius Frasincar	Erasmus University Rotterdam, The Netherlands
Dr. Wu He	Old Dominion University, USA
Dr. Marjan Hericko	University of Maribo, Slovenia
Dr. George Karabatis	University of Maryland, Baltimore County, USA
Dr. Dario Liberona	University of Santiago, Chile
Dr. Remy Magnier-Watanabe	University of Tsukuba, Tokyo, Japan
Dr. Victor Hugo Medina Garcia	Universidad Distrital Francisco José de Caldas, Colombia
Dr. Marja Naaranoja	Vaasa University of Applied Sciences, Finland
Dr. Paulo Novais	University of Minho, Portugal
Dr. Takao Terano	Tokyo Institute of Technology, Japan
Dr. Derrick Ting	National University of Kaohsiung, Taiwan
Dr. Lorna Uden	Staffordshire University, UK
Dr. Michael Vallance	Future University Hakodate, Japan

Dr. William Wang	Auckland University of Technology, New Zealand
Dr. Leon S.L. Wang	National University of Kaohsiung, Taiwan
Dr. Guandong Xu	Victoria University, Australia
Dr. Ales Zivkovic	University of Maribo, Slovenia

Organization Committe

Dr. Juan M. Corchado (Chair)	University of Salamanca, Spain
Dr. Javier Bajo (Vice-chair)	Pontifical University of Salamanca, Spain
Dr. Sara Rodríguez	University of Salamanca, Spain
Dr. Francisco de Paz Santana	University of Salamanca, Spain
Dr. Emilio S. Corchado	University of Salamanca, Spain
Dr. Belén Pérez	University of Salamanca, Spain
Fernando De la Prieta	University of Salamanca, Spain
Carolina Zato	University of Salamanca, Spain
Antonio J. Sanchez	University of Salamanca, Spain

Contents

Evaluation of a Self-adapting Method for Resource Classification in Folksonomies

José Javier Astrain, Alberto Córdoba, Francisco Echarte, and Jesús Villadangos

Dept. Ingeniería Matemática e Informática, Universidad Pública de Navarra, Campus de Arrosadía, 31006 Pamplona, Spain
josej.astrain@unavarra.es, alberto.cordoba@unavarra.es, patxi@eslomas.com, jesusv@unavarra.es

Abstract. Nowadays, folksonomies are currently the simplest way to classify information in Web 2.0. However, such folksonomies increase continuously their amount of information without any centralized control, complicating the knowledge representation. We analyse a method to group resources of collaborative-social tagging systems in semantic categories. It is able to automatically create the classification categories to represent the current knowledge and to self-adapt to the changes of the folksonomies, classifying the resources under categories and creating/deleting them. As opposed to current proposals that require the re-evaluation of the whole folksonomy to maintain updated the categories, our method is an incremental aggregation technique which guarantees its adaptation to highly dynamic systems without requiring a full reassessment of the folksonomy.

1 Introduction

Folksonomies are nowadays a widely used system of classifying information for knowledge representation [10]. Tags made by users provide semantic information that can be used for knowledge management. As users annotate the same resources, frequently using the same tags, their semantic representations for both tags and annotated resources emerge [12, 13, 15]. Folksonomies are based on the interaction of multiple users to jointly create a "collective intelligence" that defines the semantics of the information. Although users follow an easy and simple mechanism to classify resources, knowledge representation becomes more difficult as the volume of information increases [3]. Folksonomies are difficult to be studied due to their three-dimensional structure (hypergraph) [2, 9]. Then, different two-dimensional contexts of this information are often considered [2] (tag-user, tag-resource and tag-tag).

Folksonomies are highly dynamic systems, so any proposal should be scalable and able to adapt to its evolution. In [8], a generalization of the methods used to obtain two-dimensional projections dividing them into non-incremental aggregation methods and incremental aggregation methods is proposed. The first one includes solutions similar to those proposed in [2, 9] where the incorporation of new information to the folksonomy involves the complete recalculation of the similarity matrices. The second one includes solutions in which new annotations introduced in

© Springer-Verlag Berlin Heidelberg 2013

the folksonomy do not involve a repetition of all the calculations. Faced with these tag-centric based proposals, resource-centric approaches have been poorly studied with the aim of structuring the folksonomy resources. In this paper, we consider the improvement of the folksonomy knowledge representation by creating semantic categories, also called concepts, that group the folksonomy resources. We try to obtain the relationship between kolksonomies and ontologies obtaining the semantics from both tag and resources.

Some works in literature [1, 14] attempt to classify the resources of a folksonomy into concepts to offer improvements in user navigation. In [1] the resources of a folksonomy are classified under a set of classification concepts. These classification concepts are previously obtained through a manually search through a repository of ontologies. Once obtained the classification concepts, an algorithm classifies the tags of the folksonomy under the concepts obtained combining the co-occurrences of the tags and the results obtained after submitting certain search patterns to Google. Folksonomy resources are classified into concepts according to the tags they have been assigned. Furthermore, the authors do not propose an implementation or prototype of their proposal, so it is not possible to assess to what extent the classification aided navigation. This non-incremental method depends on the intervention of a user which defines the terms for the classification of ontologies in which resources are classified, and the use of external information sources. Authors also fail to take into account the evolution of the folksonomy and how to adapt their proposal to the incorporation of new tags and resources.

An optimization algorithm for the classification of resources under a set of concepts, using a set of predefined concepts and a set of previously classified resources, is used in [14]. In this regard, the goal of this algorithm is similar to the method described in this work, since it directly classifies the resources under concepts. However, the algorithm has some drawbacks such as: a) the categories are fixed, and their evolution are not considered; b) it requires full reassessment of the algorithm each time the folksonomy evolves since it is a non-incremental method; and c) it does not describe how resources are sorted within each concept.

This paper introduces a simple method for the automatic classification of the resources of a folksonomy into semantic concepts. The main goal is to improve the knowledge management in folksonomies, keeping the annotation method as simple as usual. Folksonomies are highly dynamic systems where new tags and resources are created continuously, so the method builds and adapts these concepts automatically to the folksonomy's evolution. Concepts can appear or disappear by grouping new resources or disaggregating existing ones and resources would be automatically assigned to those concepts. Furthermore, the method has a component-based open architecture which allows its application to folksonomies with different characteristics. It uses a reduced set of the folksonomy's tags to represent both the semantics of the resources and concepts because it requires a high classification efficiency to allow its application to real folksonomies (where the number of annotations grows very fast), and assigns automatically an appropriated name to those concepts.

The rest of the paper is organized as follows: Section 2 describes the method; Section 3 deals with the method evaluation using a real folksonomy; Acknowledgements, Conclusions and References end the paper.

2 Method Description

The method, introduced in [6], follows a component based open architecture, which allows its application to folksonomies with different characteristics. It requires a high classification efficiency to allow its application to real folksonomies, where the number of annotations grows very fast.

Given a folksonomy, the method initially creates a set of concepts where resources are grouped, and it assigns a name to each concept according to the semantic information provided by the resources grouped in each concept and their annotations. Once created the concepts, each new annotation of the folksonomy is processed updating the semantic information and adapting the concepts when necessary. The method starts with the creation of the set of representative tags (S_{rt}) and the vectorial representation of the resources of the folksonomy. The component *Representations* is in charge of these tasks. It may use different criteria to create S_{rt} like the tags more frequently used, the tags used by more users and even more. Then, each resource is assigned to subsets $R_{converged}$ or $R_{pending}$ in terms of whether they have converged or not. In order to obtain those tags that best describe the semantic of a resource, in [11], it is showed that tagging distributions of heavily tagged resources tend to stabilize into power law distributions. The component *Convergence* may use many criteria like the total amount of annotations, or the number of annotations associated to the S_{rt} set to assign the resources to $R_{converged}$.

The component *Clustering* clusters the resources of the folksonmy belonging to $R_{converged}$ in a set of concepts, generating the set of semantic concepts on which resources of the folksonomy are grouped (C) and the set of pairs (r,c) where $r \in R$ and $c \in C$, representing that resource r is grouped into the concepts c (Z). The initial clustering of the resources may follow different criteria: applying clustering techniques to the resources of the folksonomy, creating manually the classification concepts and selecting a relevant resource as seed of the classifier, or adapting some tag clustering algorithms like T-Know [1] in order to classify resources, instead of tags, under the concepts that have been previously selected. The component *MergingSplitting* analyses the concepts provided in order to evaluate the convenience of merging or splitting any of them, updating C and Z sets. It merges those concepts whose similarity values are greater than 0.75, using the cosine measure. The component *Classifier* is in charge of grouping the resources under those concepts with which they have high semantic similarity by comparing all the resources belonging to the $R_{converged}$ set with the concepts in which they are grouped in Z. The component assigns the resource to the $R_{classified}$ set according to the similarity measures between each resource and its group or keeps it on $R_{converged}$ and removes it from Z. The component *Representations* creates the vector representations for the C concepts using the Z set, and the component *Naming* assigns meaningful names to these concepts according to some criteria like the tags with higher weights.

```
1.  Representations::createS_rt()
2.  Representations::createVectors()

3.  forall $r \in R$ do
4.      if Convergence::hasConverged($r$) then
5.          assign $r$ to $R_{converged}$
6.          else assign $r$ to $R_{pending}$
7.      endif
8.  endforall

9.  Clustering::create($R_{converged}$)
10. MergingSplitting::process($C,Z$)

11. forall $r \in R_{converged}$ do
12.     if Classifier::isCorrectlyClassified($Z,r$) then
13.         assign $r$ to $R_{classified}$
14.         else drop $r$ from $Z$
15.     endif
16. endforall

17. Representations::createConceptVectors($C,Z$)
18. Naming::process($C,Z$)

19. while true do

20.     tagging = wait(FolksonomyEvolution)

21.     if not Representations::inS_rt($t$) then
22.         continue
23.     endif

24.     Representations::updateVectors($Tagging$)

25.     if tagging.action = create then
26.         if $r \in R_{pending}$
                and Convergence::hasConverged($r$) then
27.             assign $r$ to $R_{converged}$
28.         endif
29.         if $r \in R_{converged}$ then
30.             Classifier::classify($Z,r$)
31.             Representations::updateConceptVector($Z,r$)
32.         endif
33.     endif

34.     if RecalculationCondition::check() then
35.         Representations::updateS_rt()
36.         Clustering::update($C,Z$)
37.         MergingSplitting::process($C,Z$)
38.         Representations::updateVectors($C,Z$)
39.         Naming::process($C,Z$)
40.     endif

41. endwhile
```

Fig. 1 Method algorithm

At this point, the method has built $R_{pending}$, $R_{converged}$, $R_{classified}$, C and Z sets from the folksonomy, so it provides a set of concepts that group folksonomy resources based on their semantics. Once created these concepts, the method self-adapts to the evolution of the folksonomy taking into account the new annotations made by users. The lines 19 to 41 of the pseudocode described in Fig. 1 are responsible of processing these new annotations. The method waits for a change on the folksonomy when creating or removing an annotation. This change is represented by the method as a new *Tagging* element, which contains the annotation information (user, resource and tag), and if it has been created or deleted. If the tag used in the *Tagging* does not belong to the representative set of tags (S_{rt}), *Tagging* is ignored and it expects the reception of new annotations. If this tag belongs to the S_{rt} set, the component *Representations* updates the vectorial representation of the resource. If the resource belongs to the $R_{classified}$ set the component also updates the vectorial representation of the concept in which the resource is grouped. If the *Tagging* is of type *create*, the method checks whether the resource has converged or not, and the possibility of grouping it under any existing concept. If the resource belongs to the $R_{pending}$ set, the component *ConvergenceCriterion* checks if the resource has converged after receiving the new annotation. If so, or if the resource already previously belonged to $R_{converged}$, the component *Classifier* provides the most appropriate concept for this resource. The component compares the semantic of the resource with that of each concept in C. Based on this similarity, the component assigns the resource to the $R_{classified}$ set and creates a new entry in Z, or lets the resource continue to be assigned to $R_{converged}$. If the resource is assigned to a concept, *Representations* component updates the vectorial representation of the concept with the resource information.

The method groups the resources in concepts, so that once a resource is classified it will never return to the set $R_{converged}$. Therefore, when a *Tagging* of type *delete* is received, the method does not checks if the resource has converged or whether it must continue to exist under the current concept, the method only updates the corresponding vectorial representation. In addition to gathering new converged resources into existing concepts, the method considers the information received from the new *Taggings*, updating the S_{rt} set and the existing concepts. Thus, S_{rt} and C sets may adapt to the folksonomy's evolution, performing their adaptation for example to new users' interests. Since a unique *Tagging* does not use to significantly affect the S_{rt} set or the concepts set, and this update can be quite expensive computationally, the method uses the component *RecalculationCondition* to determine when to update both concepts and S_{rt}. The recalculation may be performed considering many criteria, for example, after a certain number of processed *Taggings*, after a given time period, or whenever a resource or a set of resources are classified. When the component determines the convenience of performing the recalculation, in a first step the S_{rt} set is updated taking into account its establishment criteria using component *Representations*. It then uses the *Clustering* component to update the existing concepts (C) and the resources grouped in them (Z). The component *MergingSplitting* reviews these concepts creating, splitting them when necessary. Once obtained the elements C and Z, *Representations* updates the concept representation vectors, and *Naming* assigns a name to each one of the concepts. Upon the completion of these tasks, the method returns to stand waiting for the arrival of new *Taggings* to the folksonomy for their processing.

3 Method Evaluation

This section is devoted to evaluate experimentally the proposed method using data retrieved from Del.icio.us. The method creates automatically a set of concepts from an existing folksonomy and groups under these concepts the resources of the folksonomy, according to their semantics. As the folksonomy receives new annotations, the method groups new resources, takes into account new relevant tags, and adapts the existing concepts.

With the aid of a page scraper we have collected a set of 15,201 resources, with 44,437,191 annotations, 1,293,351 users and 709,657 tags from Del.icio.us (15th-30th September, 2009)[1]. Those annotations, depicted in Table 1, concern a period time from January 2007 to September 2009. We use annotations prior to 2009 to simulate an initial state of the folksonomy in order to create the initial concepts. The rest of annotations correspond to January to September 2009 and they are used to simulate the folksonomy evolution by means of *Taggings* elements of type *create*. Table 2 summarizes the information concerning the initial folksonomy (t_0) and its state after including the *Tagging* elements concerning the period Jan.-Sept. (t_1 to t_9).

[1] http://www.eslomas.com/publicaciones/KMO2012/

Table 1 Annotation distribution

Year	Annotations	Year	Annotations	Year	Annotations
1998	2	2002	299	2006	3,140,591
1999	3	2003	628	2007	7,237,129
2000	7	2004	52,345	2008	13,753,922
2001	12	2005	719,216	2009	19,533,037

Table 2 Users, tags and resources in each subset of the experiment

	Increment of the number of elements				Aggregated values			
t_i	annotations	users	resources	tags	annotations	users	resources	tags
t_0	24,904,154	972,695	12,117	489,125	24,904,154	972,695	12,117	489,125
t_1	1,704,682	35,480	342	23,581	26,608,836	1,008,175	12,459	512,706
t_2	1,811,331	37,240	353	23,066	28,420,167	1,045,415	12,812	535,772
t_3	2,179,539	40,672	407	26,101	30,599,706	1,086,087	13,219	561,873
t_4	2,153,461	34,603	336	24,187	32,753,167	1,120,690	13,555	586,060
t_5	2,230,512	33,078	391	24,919	34,983,679	1,153,768	13,946	610,979
t_6	2,304,614	33,959	348	24,460	37,288,293	1,187,727	14,294	635,439
t_7	2,437,317	34,137	345	24,951	39,725,610	1,221,864	14,639	660,390
t_8	2,617,998	36,438	368	26,332	42,343,608	1,258,302	15,007	686,722
t_9	2,093,583	35,049	194	22,935	44,437,191	1,293,351	15,201	709,657

The method is configured for the experimentation using the following components. The comparison between resource and concepts vectors has been performed using the cosine measure. In the following we describe the components used to configure the method. The component *Representations* considers a S_{rt} set built using those tags with at least 1,000 annotations. The component *Convergence* fixes the convergence criterion to 100 annotations [7]. The component *Classifier* uses the method presented in [5] to classify the resources under the most similar concepts. In the evolution task, the classifier applies for a given resource each time it reaches a multiple of 50 annotations. This component has been configured to take into account the two most similar concepts (c_i, c_j) to each resource (r_i), and classify only the resource when the difference between the resource and these concepts is greater than a minimum threshold value of 0.10 ($|sim(r_i,c_i) - sim(r_i,c_j)| \geq 0.10$). As proved in [5], the method is able to classify the resources providing a high precision. The component *Recalculation* recalculates monthly both sets S_{rt} and C (after each t_i $(i : 1..9)$). The component *Clustering* uses a k-means algorithm, determining the k value by the expression $k = \sqrt{\frac{n}{2}}$, being n the number of resources in $R_{converged}$ at the concepts creation and in $R_{classified}$ when recalculating. Thus, the number of concepts can grow as the folksonomy evolves. At the initial concepts creation, the initial centroids are randomly defined since any a-priori knowledge is not considered. When recalculating, the representation vectors of C concepts are used to define the initial centroids, and if $k \geq |C|$, then some resources are randomly selected to define the $k - |C|$ new clusters. The implementation of the algorithm is performed in a distributed way where the calculus of the number of cluster changes at each iteration is performed over different PCs of a cluster using a task queue manager (Gearman). We use the k-means method instead of hierarchical techniques because we want to provide a concept cloud (see Figure 2) with a similar user experience to tag clouds. Component *MergingSplitting* merges using the cosine measure those concepts whose similarity values are greater than 0.75, once the *Clustering* component obtains C and Z sets.

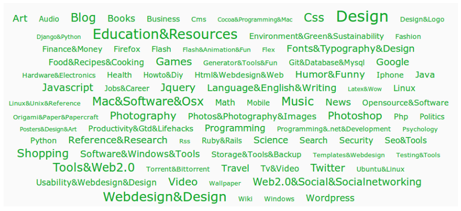
Art Audio Blog Books Business Cms Cocoa&Programming&Mac Css Design Design&Logo
Django&Python Education&Resources Environment&Green&Sustainability Fashion
Finance&Money Firefox Flash Flash&Animation&Fun Flex Fonts&Typography&Design
Food&Recipes&Cooking Games Generator&Tools&Fun Git&Database&Mysql Google
Hardware&Electronics Health Howto&Diy Html&Webdesign&Web Humor&Funny Iphone Java
Javascript Jobs&Career Jquery Language&English&Writing Latex&Wow Linux
Linux&Unix&Reference Mac&Software&Osx Math Mobile Music News Opensource&Software
Origami&Paper&Papercraft Photography Photos&Photography&Images Photoshop Php Politics
Posters&Design&Art Productivity&Gtd&Lifehacks Programming Programming&.net&Development Psychology
Python Reference&Research Rss Ruby&Rails Science Search Security Seo&Tools
Shopping Software&Windows&Tools Storage&Tools&Backup Templates&Webdesign Testing&Tools
Tools&Web2.0 Torrent&Bittorrent Travel Tv&Video Twitter Ubuntu&Linux
Usability&Webdesign&Design Video Wallpaper Web2.0&Social&Socialnetworking
Webdesign&Design Wiki Windows Wordpress

Fig. 2 Concept cloud

And finally, component *Naming* assigns a name to each concept according to the most relevant tags of its vector. When the weight of several tags is greater than the 50% of the weight of the most relevant tag, the name of the concept is obtained through the concatenation of those tags after verifying, by means of the Levenshtein distance, that tags are not syntactic variations of other previous tags. So, a concept in which the two most relevant tags are *php* (weight 127,427) and *programming* (weight 39,743), is named “Php”.

Besides Gearman, Memcached has been used as a cache system in order to reduce the number of accesses to the database to obtain resources, concepts and representation vectors. The employed hardware consists of four commodity PC with Intel Core 2 Duo processors at 2.13 GHz, and 2GB of RAM memory. In order to perform the distributed tasks, 8 processes (2 on each PC) have been executed to perform the processing of the K-means slices, and 24 processes have been executed to process the *Tagging* processing tasks.

Table 3 summarizes the information concerning the evolution of the number of tags, resources and concepts, from t_0 to t_9. Regarding the tags and the S_{rt} set, it shows the evolution of the number of tags of the folksonomy and the number of tags in S_{rt}, and the ratio between them. One can note that the number of tags in S_{rt} increases as the number of annotations in the folksonomy does. The increment in the number of annotations carries with a higher number of tags exceeding the threshold of 1,000 annotations required to be part of S_{rt}. Lets note that: i) the method represents the semantics of the resources with less than 0.40% (1,939/489,125) of the existing tags; and ii) this value decreases slowly as more annotations arrive to the folksonomy (up to 0.38% after processing t_9). This makes the cost of the method, in terms of space and process, significantly lower than the required when considering all the tags of the folksonomy to represent the semantics of the resources and their concepts associated. Table 3 shows that the number of classified resources increases as the number of resources of the folksonomy does. As folksonomy evolves with new annotations, some of the $R_{pending}$ resources converge and pass to $R_{converged}$ set, while other resources receive enough annotations to determine an adequate concept, passing to $R_{classified}$. Regarding concepts, Table 3 also shows the evolution in the k value used by the *Clustering* component, and the number of concepts created at each recalculation. When recalculating, *Clustering* component tries to create a number of k concepts considering the number of $R_{classified}$ resources, however, some of these concepts may be merged by *MergingSplitting* component when their similarity are greater than the defined threshold (0.75). Note that both k and $|C|$ values gradually increases after each recalculation. Although in this experiment the number of concepts increases, the method allows the creation, splitting and merging of concepts, so this number may also decrease. In this experiment the clustering is based on k-means, with k depending on the number of resources, causing the maintenance or increase of the number of concepts. The number of concepts can decrease only if, after clustering, the *MergingSplitting* component finds that there are two or more concepts with a similarity degree greater than 0.75.

Table 3 Adaptation of the method as the folksonomy evolves

		Tags			Resources		Concepts	
t_i	$\lvert T\rvert$	$\lvert S_{rt}\rvert$	$\frac{\lvert S_{rt}\rvert}{\lvert T\rvert}$	$R_{pending}$	$R_{converged}$	$R_{classified}$	$\lvert C\rvert$	k
t_0	489.125	1.939	0,40%	337	3.098	8.682	75	77
t_1	512.706	2.037	0,40%	289	3.184	8.986	76	78
t_2	535.772	2.124	0,40%	305	3.190	9.317	77	80
t_3	561.873	2.204	0,39%	282	3.226	9.711	78	81
t_4	586.060	2.286	0,39%	269	3.237	10.049	78	82
t_5	610.979	2.362	0,39%	250	3.277	10.419	79	83
t_6	635.439	2.437	0,38%	225	3.300	10.769	80	84
t_7	660.390	2.526	0,38%	206	3.033	11.400	81	85
t_8	686.722	2.616	0,38%	139	2.915	11.953	82	86
t_9	709.657	2.716	0,38%	43	2.917	12.241	83	87

Analysing the creation of concepts and the evolution in the number of resources grouped in each concept, one can note that the creation of certain concepts produces a decrease in the number of resources grouped in other concepts. For example, after processing the *Tagging* set t_9, the new annotations received in the transition from the eighth to ninth iteration lead the concept $Business\&Finance\&Money$ be split into two concepts: $Business$ and $Finance\&Money$. The initial concept ($Business$), which previously brought together 215 resources, now groups 118 resources. The new concept created groups 119 resources, of which 110 (92 %) were previously grouped under $Business\&Finance\&Money$. In most cases, the number of resources classified under each concept grows as the number of annotations of the folksonomy does, but in some cases, this number may decrease. This decrease occurs either when the concept under which resources were classified is split either when the arrival of new annotations allow a better definition of the classification concept. As occurs with $Web2.0\&Social\&Socialnetworking$, in which the number of classified resources decreases from 244 to 230 in the transition from iteration 8 to 9. In this transition, 220 resources remain classified under this concept, 24 of them are reclassified under other existing concepts ($Video$, $Business$, $Twitter$, $Blog$, $Tools\&Web2.0$ and $Generator\&Tools\&Fun$), 6 new resources are classified under this concept and the remaining 4 resources were previously classified under other concepts.

Table 4 Number of resources classified under the eight first concepts

Concept		Statistics				Iteration									
		Avg	Dev	Max	Min	0	1	2	3	4	5	6	7	8	9
1	Hardware&Electronics&Technology	4.2	64	72	60	60	61	62	63	64	64	66	69	71	72
2	Jobs&Career&Job	85	6.4	95	75	75	77	81	83	84	84	88	90	92	95
3	Books&Literature	171	13.0	192	152	152	156	161	166	169	172	177	182	185	192
4	Video&Web2.0	214	26.9	252	175	175	183	192	201	206	221	225	237	248	252
5	Html&Webdesign&Web	85	9.2	103	75	75	77	80	80	80	81	85	88	98	103
6	Linux&Opensource	116	6.4	125	108	108	109	110	112	115	117	118	122	125	125
7	Art&Design	198	7.0	208	189	189	190	195	193	194	200	204	205	206	208
74	Howto&Diy	81	8.6	93	70	82	70	72	72	75	83	86	88	92	93

Table 5 Evolution of the resources classified

Number of changes	Initial iteration t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	Total
1	320	25	29	17	11	14	11	26	22	0	475
2	79	2	1	3	2	3	0	1	0	0	91
3	2	0	1	0	0	0	0	0	0	0	3
4	1	0	0	0	0	0	0	0	0	0	1

Table 4 shows the number of resources classified under some concepts. In the same way, the creation of the concept *Origami*, which groups 16 resources, occurs after processing the set t_1 and its resources come from *Howto& Diy*, *Art&Design* and the rest are resources classified during the processing of t_1 and therefore were not grouped into any concept after t_0. It has been found therefore that the origin of the resources of the concepts created in the evolution presents a semantic relationship to the concepts created. This indicates that as the folksonomy receives new annotations, the method is able to group all the related resources and to evolve the concepts so as to adapt to user interests: e.g., resources that were initially grouped into *Howto&Diy*, representing pages with instructions on how to do things with the appearance in the folksonomy of more information on origami causes the method to create a specific concept for this topic, bringing together new and existing resources.

Table 5 shows the evolution of the resources classified. The left side of the table shows that 11,671 of the 12,241 resources retain their original classification along the evolution of the folksonomy, 475 of the resources change their concept of classification once, 91 of them change twice, 3 of them make it thrice, and only 1 makes it 4 times. The right side shows the distribution of the number of changes among classification concepts experienced by resources according to the iteration in which they are introduced. Of the set of resources introduced in the initial iteration which change their classification concept along the evolution of the folksonomy, 320 of them make it once, 79 of them change their classification concept twice, and so on. An analysis of the 83 classification concepts involved in this process shows that 75 of them appear in the initial iteration, and the rest appear one for each iteration, except in the fifth iteration, which does not introduce any new concept.

Regarding the performance of the method, the time spent in each stage has been registered. Table 6 shows the time needed to create the initial concepts (lines 1-18 of the method described in Fig. 1).

Table 6 Creation costs for the initial concepts

Action	Time (minutes)
S_{rt} creation	1.10
Vectorial representation creation	22.83
Convergence verification	0.08
Clustering (k=76)	192.21
MergingSplitting	0.98
Assignation to $R_{classified}$	61.62
Creation of concept' vectors	0.53
Naming	0.01

Table 7 Time spent in evolution

		Time (minutes)	
t_i	Number of taggings	Tag. Processing	Recalculation
t_1	1,704,682	28.23	23.47
t_2	1,811,331	31.24	24.35
t_3	2,179,539	41.98	48.91
t_4	2,153,461	43.12	29.18
t_5	2,230,512	40.36	56.45
t_6	2,304,614	41.12	27.39
t_7	2,437,317	41.84	81.34
t_8	2,617,998	32.65	27.26
t_9	2,093,583	37.34	63.12

Table 7 shows the time spent processing the *Tagging* elements of t_1 to t_9 sets (lines 19 to 41 in Fig. 1). Regarding the *Tagging* elements processing, the method has processed them with an average throughput of 965.74 *Tagging* elements per second, which represents an average of 40.24 *Tagging* elements per second and tagging processing worker (24 workers, threads annotating concurrently). These results show the applicability of the method in real systems, processing the new annotations to adapt the classification concepts. Regarding the time spent in the recalculation, it is quite variable, depending on the number of iterations of k-means algorithm. However, the time spent by the *Clustering* component when recalculating is much lower than the time spent at the first clustering (t_0), because the representation vectors of C are used as initial centroids, while at the first clustering the centroids are randomly created. We are now working on the use of different clustering techniques, including hierarchical clustering techniques, implementing the *Clustering* component over Apache Mahout, which implements a Framework MapReduce[4] allowing the usage of different clustering techniques.

In order to validate the quality of the classification of resources obtained, we have evaluated the 12,241 resources considered with the aid of 102 computer science students (advanced and regular internet users) at the Universidad Pública de Navarra during the course 2010-2011. Each reviewer has evaluated a subset of the resource set, ensuring that each resource has been evaluated by five different reviewers. Each reviewer has evaluated its subset of resources after the initial classification, and then those new resources and those whose category of classification has changed along the different recalculations. Reviewers evaluated, for each of the resources, how well resources are classified under their category of classification, quantifying this value between 1 and 5, meaning (1) a very poor classification, (2) a poor classification, (3) a reviewer indecision, (4) a good classification and (5) a very good classification. Reviewers considered 106 resources *very poor* classified (1%), 259 *poor* classified (2%), 4,492 *good* classified and 6,313 *very good* classified. The reviewers were hesitant with 971 resources (8%).

4 Conclusions

We have proposed, implemented and analysed a simple and incremental method for the automatic and semantic creation of concepts to group the resources of a folksonomy, in order to improve the knowledge management in folksonomies, without changing the way users make their annotations. The method automatically creates these concepts and adapts them to the folksonomy evolution over time, grouping new resources and creating, merging or splitting concepts as needed. It is an incremental-aggregation technique that adapts to the folksonomy evolution, without requiring to re-evaluate the whole folksonomy.

The method uses a small subset of tags, the set of more representative tags (S_{rt}), in order to apply it to real folksonomies and their evolution without adversely affecting its performance. Furthermore, the method is based on a component based open architecture. This allows its application to folksonomies with different features and needs, and a useful way to assign names to the classification concepts.

The semantic information assigned to the resource by their annotations allows the automatic classification of the resources of a folksonomy under classification concepts without requiring the intervention of human experts. We have experimentally validated, with the aid of human experts, that the method is able to create automatically these concepts and adapt them to the folksonomy evolution over time, classifying new resources and creating new concepts to represent more accurately the semantic of the resources.

Acknowledgements. Research partially supported by the Spanish Research grants TIN2011-28347-C02-02, TIN2010-17170 and INNPACTO-370000-2010-36. The authors also wish to acknowledge their collaboration to students who have validated the proposal.

References

1. Abbasi, R., Staab, S., Cimiano, P.: Organizing resources on tagging systems using t-org. In: Bridging the Gap between Semantic Web and Web 2.0 (2007)
2. Cattuto, C., Benz, D., Hotho, A., Stumme, G.: Semantic Grounding of Tag Relatedness in Social Bookmarking Systems. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 615–631. Springer, Heidelberg (2008)
3. Chi, E.H., Mytkowicz, T.: Understanding navigability of social tagging systems. In: Conference on Human Factors in Computing Systems (2007)
4. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)

5. Echarte, F., Astrain, J.J., Córdoba, A., Villadangos, J., Labat, A.: Acoar: a method for the automatic classification of annotated resources. In: Proc. of the 5th Int. Conference on Knowledge Capture, pp. 181–182. ACM, New York (2009)
6. Echarte, F., Astrain, J.J., Córdoba, A., Villadangos, J., Labat, A.: A self-adapting method for knowledge management in collaborative and social tagging systems. In: 6th Int. Conf. on Knowledge Capture, pp. 175–176. ACM, New York (2011)
7. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems. Journal of Information Science 32(2), 198–208 (2006)
8. Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G.: Evaluating similarity measures for emergent semantics of social tagging. In: Proc. of the 18th International Conference on World Wide Web, pp. 641–650. ACM, New York (2009)
9. Mika, P.: Ontologies are us: A unified model of social networks and semantics. Web Semantics: Science, Services and Agents on the World Wide Web 5(1), 5–15 (2007)
10. Peters, I.: Folksonomies: indexing and retrieval in Web 2.0. Knowledge & Information: Studies in Information Science. De Gruyter/Saur (2009)
11. Robu, V., Halpin, H., Shepherd, H.: Emergence of consensus and shared vocabularies in collaborative tagging systems. ACM Transactions on the Web 3(4), 1–34 (2009)
12. Staab, S.: Emergent semantics. IEEE Intelligent Systems 17(1), 78–86 (2002)
13. Steels, L.: The origins of ontologies and communication conventions in multi-agent systems. Autonomous Agents and Multi-Agent Systems 1(2), 169–194 (1998)
14. Yin, Z., Li, R., Mei, Q., Han, J.: Exploring social tagging graph for web object classification. In: Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 957–966. ACM, New York (2009)
15. Zhang, L., Wu, X., Yu, Y.: Emergent Semantics from Folksonomies: A Quantitative Study. In: Spaccapietra, S., Aberer, K., Cudré-Mauroux, P. (eds.) Journal on Data Semantics VI. LNCS, vol. 4090, pp. 168–186. Springer, Heidelberg (2006)

Emerging Concepts between Software Engineering and Knowledge Management

Sandro Javier Bolaños Castro[1], Víctor Hugo Medina García[1], and Rubén González Crespo[2]

[1] District University "Francisco José de Caldas", Bogotá, Colombia
[2] Pontificial University of Salamanca, Madrid, Spain
{sbolanos,vmedina}@udistrital.edu.co,
ruben.gonzalez@upsam.net

Abstract. Software Engineering (SE) uses different theories to empower its practices. One such theory is Knowledge Management (KM), which provides an important conceptual heritage. Our proposal establishes emerging concepts that enrich SE from KM. All these concepts are in between knowledge and software, hence we call them Softknowledge (SK), y Hardknowledge (HK); they constitute Knowledgeware (KW). In this paper we emphasize the intentionality that pertains to these concepts, which is a fundamental characteristic for the development, maintenance, and evolution of software. Additionally, we propose a nurturing environment based on the present proposal.

1 Introduction

Since the term SE was coined [1], there has been constant crisis within this discipline. This crisis can be seen when observing the discouraging figures reported by official bodies like the Standish Group [2], where high percentages of failure on the projects conducted are reported. Aiming at improving such a bleak situation, a variety of software development processes have been proposed, ranging from code and fix [3] to Waterfall [4], Spiral [5], V [6], b [7], RUP [8], among others. On the other hand, methodologies such as XP [9], Scrum [10], Crystal [11], ASD [12] have been proposed to address the development of practices, values and principles [13]. It seems that knowing the way (how), the people (who), the place (where) and the time (when), that contribute to the development, the processes and the methodology around a problem is a suitable roadmap; however, discouraging reports continue to appear.

The present paper points in a different direction, namely knowledge management and it attempts to reduce failure of the software projects. Although this approach was studied already [14], [15], [16], emerging concepts in between the theories of Software Engineering and Knowledge Management have not been proposed to strengthen the understanding and the solutions to some of the most relevant problems encountered in Software Engineering.

2 Problem Statement

Although Software Engineering has been empowered from Knowledge Management, the work done so far has adopted concepts that, from the perspective of Knowledge Management, favor Software Engineering. One of these concepts is what is known as *tacit knowledge* [17]; however, when adopting a concept, not only are the solutions brought in but also the problems associated to the framework within the providing discipline of origin. Even though the classification of both tacit knowledge and explicit knowledge [18] together with its processes framework (SECI) [19] represent a good approximation to many of the problems found in collaborative work – which is typical of software projects such as: the strengthening of knowledge generation [20], knowledge gathering [21], knowledge exchange [22] and knowledge co-creation [23] processes – it is clear that engineers always end up facing the same advantages and disadvantages that have already been identified for the same processes in the field of KM.

The tendency of researchers to narrow the gap between two theories as a means of improving knowledge frameworks causes harmful side effects, which are eventually identified in most cases. There is a special and apparent similarity between software engineering and knowledge management, but you need to check the most favorable concepts carefully and find mechanisms beyond the mere adoption and adaptation of these ideas.

We believe that, more than doing a theory transfer, it is necessary to generate emerging concepts.

3 Knowledgeware: The Missing Link between Knowledge and Software

Software is a product obtained from intellect [24], it is an extension of our thoughts [25]. When software products are made, there are as many variables as people participating in the development process – "Conway's law" [26], [27]. It could also be claimed that software is knowledge; however, such an statement is too straightforward and it would be reckless not to acknowledge knowledge as intelligence [28], an ability [29], as states of the mind [30], beliefs, commitments, intentions [18] among other features; and although software is pervaded with our reasoning, that reasoning is constrained by the conditions of the programming language and programming paradigms [31], and also by the intended purposes [32] of the documents supporting the software product. The solution to this situation is knowledgeware KW; this new species can be found between knowledge and software, and it takes its traits from both concepts.

Given that, from the perspective of knowledge management, there is a clear distinction between tacit knowledge and explicit knowledge and such a distinction points to the possibility of coding; likewise, in the case of KW, we propose to have a distinction between what we call softknowledge SK and hardknowledge HK. From the point of view of coding, tacit knowledge is orthogonal to explicit knowledge. While tacit knowledge resides in people, explicit knowledge can be stored using physical media, typically IT media. One of the most recurrent concerns is perhaps the way in which organizations gather knowledge and manage it, and so different approaches have been proposed [33], [34], [35], [36], [37], [38]. We consider that leaving tacit knowledge to the world of people is the most suitable approach, our concern should actually focus on the construction of a bridge to join tacit knowledge and software; this bridge is what we regard as SK. Likewise, there should also be a bridge between explicit knowledge and software, which is what we regard as HK.

Knowledge resides in peopleware PW [39], SK and HK reside in KW, and ultimately software resides in hardware. Knowledge management analyses PW from the point of view of the organization; among some of these frameworks we find: organizational knowledge management pillar frameworks [33], intangible asset frameworks [34], intellectual asset model [35], knowledge conversion frameworks [18], knowledge transfer model [36], knowledge management process model [37], and knowledge construction frameworks [38], among others. Regarding these models and frameworks, a lot of effort has gone into defining different types of processes that have an impact on the knowledge of the people who make up the organization. Likewise, within software engineering, the problem of conducting a project has been addressed, where projects themselves are based on organizations consisting of people who must assume the responsibility of a given process in order to obtain software. While in knowledge management processes lead to knowledge, processes in software engineering lead to software. We propose that in order to find a path from knowledge, through KW, up to software, it is necessary to conduct both a traceability process and a representation process. Fig. 1.

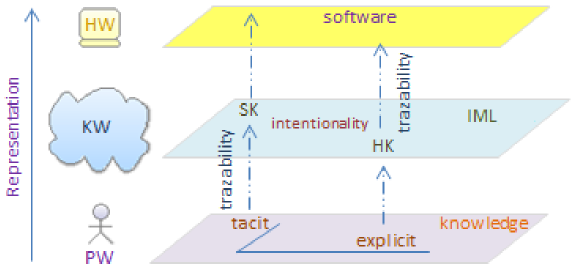

Fig. 1 Knowledge Model

3.1 *Traceability*

Software traceability [40] is a powerful mechanism that allows the construction of conceptual continuity. Such continuity must permit going back on every step taken in order to understand the origin of concepts [41]. The traceability we propose goes from tacit knowledge to SK, and from there on it goes all the way to end up becoming software; it also goes form explicit knowledge to HK, and form there on it ends up in software.

3.2 Representation

In order to make traceability visible, we propose carrying out representation through languages. If traceability remains over the path tacit-knowledge-to-SK-to-software, we suggest exploiting the advantages of widely-recognized modeling languages, namely archimate [42], UML [43], [44], SPEM [45], or exploiting notations such as BPMN [46] supported by international documents [47]. We also propose the creation of ontological entities supported by languages such as OWL [48]. We believe that software development, when oriented to the creation of models capable of intentional perspectives, can capture certain descriptions where software approximates the knowledge wherein it was created. If traceability also remains on the path explicit-knowledge-HK-software, we suggest exploiting the advantages of having a key between modeling and programming languages, whose perspectives and transformations [49] lead the way from the models language to the programming language.

We coined the concept of Intentional Modeling Language IML, fig. 1, whose additional characteristic over conventional modeling languages will be to provide representation mechanisms that allow expressing the models extended semantics. Some ways to achieve this might be found in the construction of an enriched profiles vocabulary that helps to tinge the models. It is possible to obtain these mechanisms as extension mechanisms in languages like UML. Fortunately, from the perspective of the programming language-transformation-model, this area has been widely developed e.g. MDA [50].

3.3 Intentionality

Most of the problems associated to software lie in the complexity introduced not only by source codes but also by the large volumes of documentation supporting such codes, not to include the typical risks of both coding and documents falling out-of-date. In an attempt to develop and maintain software, engineers often resort to keeping an artifact logbook, where specifications, architectural and design models, user manuals and other records can be found. However, it is by no means an easy task to deal with a product that has been expressed using a language that is intended for a machine as well as using other languages understood by humans. The most considerable difficulty is that the type of knowledge resulting from abstraction, which is expressed in documents and software artifacts, does not easily reflect the actual intention of such creation - of course it is very difficult to capture subjective expressions –; however, it should be possible at least to capture an approximate description. KW was defined as a subjective expression that contains knowledge and that allows itself to be captured through a graphical or textual description by using modeling and programming languages that can incorporate intentional expression mechanisms. Even in art, where so many different emotions are awaken, the piece of art itself provides information about its creation conditions and even about its creator; the piece of art is pervaded by the artist’s intentions. Of course in