FRAMEWORK FOR TAGGING SOFTWARE IN WEB APPLICATION

Karan Gupta[a], Anita Goel[b]

[a]Department of Computer Science, University of Delhi, Delhi, India
guptkaran@gmail.com
[b]Department of Computer Science, Dyal Singh College, University of Delhi, New Delhi, India
agoel@dsc.du.ac.in

ABSTRACT

Tagging is included in web application to ease maintenance of large amount of information stored in a web application. With no mention of requirement specification or design document for tagging software, academically or otherwise, integrating tagging software in a web application is a tedious task. In this paper, a framework is presented for integrating tagging software into a web application. The framework is for use during different stages of software development life cycle. The requirement component of framework presents a weighted requirement checklist that aids the user in deciding requirement for the tagging software in a web application, from among mandatory and optional requirements. The design component facilitates the developer in understanding the design of existing tagging software, modifying it or developing a new one. Also, the framework helps in verification and validation of tagging software integrated in a web application.

Keywords

Web Application, Tagging Software, Integration, Framework

1. INTRODUCTION

In today's world, web applications use tagging software to aid in maintenance of large amount of stored information. The integration of tagging software in a web application allows the web application to easily categorize as well as classify information and also improve searching of information. Tagging software allows its users to add keywords to a resource. Resource can be of different types like video, audio, blog, books etc. Tags belonging to a resource can describe the resource; define its type, its use, pros and cons or something entirely different.

Due to absence of a formal design document or requirement specification for tagging software, tagging functionality is integrated on-the-fly depending on whims and fancy of the developer and stated requirements of a web application. The web application owner may use free available tagging code, or adapt a freely available tagging code or write a completely new code for tagging. Generally, free tagging code is integrated after adapting it according to the needs of a web application. The code is altered to match the look and feel of a web application.

In this paper, the focus is on creating a framework that helps the user during integration of tagging software into a web application. The framework aids the user during requirement elicitation and specification based on the need of the web application. It also helps the developer in understanding structure of the tagging software and adapting or developing tagging software based on the user's requirement.

Here, a framework of tagging software is presented for integration of tagging software in a web application. The work by Gupta et al. [1] forms basis for the structural design of the tagging software and weighted requirement checklist for easing integration of tagging software. In the framework, there are four components – (1) Tagging Requirement, (2) Tagging Design, (3) Tagging Development, and (4) Tagging Test. Each component has a specific task like Tagging Requirement component performs the task of requirement generation. Similarly, the design component performs the task of generation of design. These components interact with the web application owner and the developer to generate and integrate the tagging software into web application. Each component is divided into sub-component so as to ease the completion of its task.

The framework, presented here, is used by both the developer and the owner of web application during integration of tagging software into a web application. The developer uses the framework to understand the structure of the tagging software. Moreover, the developer also gets to understand interaction among the different components of the tagging software. The owner of web application gets to know the different kind of users accessing the tagging software as well as the different kind of features provided by the tagging software. The owner of the web application is able to select these requirements.

The framework can be used during the various phases of the software development process. During the requirement phase, the framework helps in defining the requirements of the tagging software. In the design phase, the framework is used for outlining the design of the tagging software. Moreover, the framework can be used during the testing phase for verifying as well as validating the tagging software.

In this paper, section 2 explains the background for our framework. The framework is described in section 3. Section 4 lists the limitations. This is followed by the related work in section 5. Section 6 states the conclusion.

2. BACKGROUND

In web applications, use of tagging functionality facilitates categorization and classification of information. Tagging software allows users to apply keywords to resources like blog, music etc. The resource to which the tag is applied may be uploaded by a web application itself or by the user. Each resource has specific features that identify a resource, like, a video has a date of creation while a book has a publish year.

In our earlier paper [2], we have identified different type of users that access the tagging software in a web application. Also, we have identified the requirements of the tagging software and present them as a weighted requirement checklist in [1]. Using the weighted requirement checklist, the web application owner can select and define what features are to be made available in the tagging software. The design of tagging software for web application is defined by us in

[1]. The developer can use the structural design to find components that are required for providing the required features. In the following subsections, we present an overview of the kind of users accessing the tagging software, the weighted requirement checklist and the structural design.

2.1 Users of the Tagging Software

In [2], Gupta et al. identify the types of users based on their interaction with the tagging software in a web application. Using the use-case based approach; three kinds of users have been identified as follows –

- Web Application is the software in which the tagging software is integrated.
- Administrator is any person performing task of maintaining the tagging functionality.
- Visitor is a user who uses the tagging software in a web application.

Of the three users listed above, the visitor has multiple levels of flexibilities. The web application chooses the level of flexibility to be provided to a visitor accessing the web application, for tagging. The permissions that are provided to a visitor for tagging in a web application fall in three categories –

- UseTagResource (UsTR) - Visitor is able to use tags and resources only. The visitor cannot edit the resource or the tags applied to the resource.
- UseTagResource_UpdateTag (UsTR_UpT) – Visitor is able to use tags and resources and also update tags. The added or updated tags are to be approved by the administrator.
- UseTagResource_UpdateTagResource (UsTR_UpTR) - Visitor is able to use tags and resources, add tags to an already existing resource and also add a new resource and add tags to it. The administrator is given the right to moderate the changes to resources and tags.

Here, a user in the category UsTR are provided with least rights and users in UsTR_UpTR are provided with maximum rights.

2.2 Weighted Requirement Checklist for Tagging Software

Tagging software may be incorporated into web applications for different purposes. Sometimes, tagging may be integrated for providing simple features of tagging like add a resource or a tag. Others might integrate tagging for using advanced features like creating bundles of tags or writing descriptions of tags. Since the requirements of web application are different, a weighted requirement checklist has been generated by Gupta in [1]. The generated weighted requirement checklists are used for selecting operations of the tagging software, for inclusion in a web application.

Weight – Features of tagging software are given different level of importance by its users. Features like add tag, delete tags from resource are mandatory in nature. On the other hand, features like describing tags or subscribing to a resource can be optional for the user of tagging software. Three categories defined for describing the different level of importance of a feature in the tagging software –

- Required-Basic for specifying the highest level of importance. It is denoted by weight ‘3’.
- Optional-Basic for those that may be helpful in the software but are not a necessary requirement. It is denoted by weight ‘2’.
- Optional-Advanced specify the lowest level of importance, and are assigned weight ‘1’.

The weighted requirement checklist for tagging software consists of three checklists, namely, (1) Tagging Home, (2) Tagging Dashboard and (3) Tagging Parameters. Tagging Home contains operations provided to a visitor of the tagging software. This checklist allows the selection of operations for the visitor of the tagging software. Tagging Parameter consists of operations available to the web application. The checklist helps the web application to identify the parameters and their settings that are to be included in a web application during the requirement phase. Tagging Dashboard checklist contains tasks performed by the administrator in the tagging software in a web application. The checklist allows determination of the functionality to be included in the dashboard. Table 1 shows a small part of Tagging Home Weighted Requirement Checklist.

Table 1. Portion of Weighted Checklist for Component - Tagging Home

Entity		**Sub-entity**		**Operations**	
Name	W	Name	W	Name	W
Resource View	3	View	3	View a resource, Details - Title, Resource, Features*, Tags	3
				Details – Date	2
Research Search	2	Search	3	All Resources	3
				User Resources	1
Tag Cloud	2	View	3	On(Resource/User/System)	3
				Order(Cloud/List)	2
				On(Bundled/Unbundled/All), Usage(1/2/5)	1
Tag Search	2	Search	3	Text Box	3

The three weighted checklists are used during the selection of features for the tagging software. In the next sub-section, a structural design of the tagging software is presented. The structural design eases the understanding of the tagging software.

2.3 Structural Design for Tagging Software

The two building blocks of tagging software are - Resource and Tags [3]. The structural design presented in [1], is based on these two building blocks. The design displays the interaction of the visitors of different permission levels as well as the administrator with the tagging software. For the purpose of structural design, the entities for resource and tag have been identified as follows –

- Resource - Resource Update Single, Resource Update Multiple, Resource Subscription, Resource Use, Resource View, Resource Search, Resource List.
- Tag - Tag Update, Tag Bundle, Tag description, Tag Subscription, Tag Search, Tag Sharing, Tag Cloud.

The identified entities form basis of the structural design. The entities are extendible in nature and can be accommodate any new feature or functionality.

The entities identified are further divided into a group of sub-entities. Sub-entities are created for designating a specific task performed within a particular entity. Sub-entities are written as an extension of entities. Each sub-entity contains a group of operation(s) performed by the sub-entity.

The Resource part of the structural design contains two kinds of users – Administrator and Visitor. The administrator can access all entities except Resource View and Resource Subscription. On the other hand, the visitor interacts with the Structural design using two level of permissions - lowest access permission (UsTR) and highest access permission (UsTR_UpTR). The tag part of structural design interacts with two kinds of users - administrator and visitor. A visitor having access permissions UsTR and UsTR_UpT interact with entities of the tag. Administrator is able to access all entities in tag except T_Search and T_Subscription.

In this section, the structural design has been described based on premise that the tagging software is to be integrated into a web application. The entities described do not have equal importance in the tagging software.

3. TAGGING SOFTWARE INTEGRATION FRAMEWORK

Tagging software is integrated into web application to handle the large amount of information stored in the web application. The integration of tagging software is carried out using ad-hoc techniques. Based on the discussion in previous section, a framework has been developed to aid integration of tagging software into web application. The framework is designed to be used during different phases of the software development. The framework is divided into components based on its use in respective software development phase. The framework consists of 4 components as follows –

1. Tagging_Requirement
2. Tagging_Design
3. Tagging_Development
4. Tagging_Test

The relation between the software development lifecycle and our framework is depicted in Figure

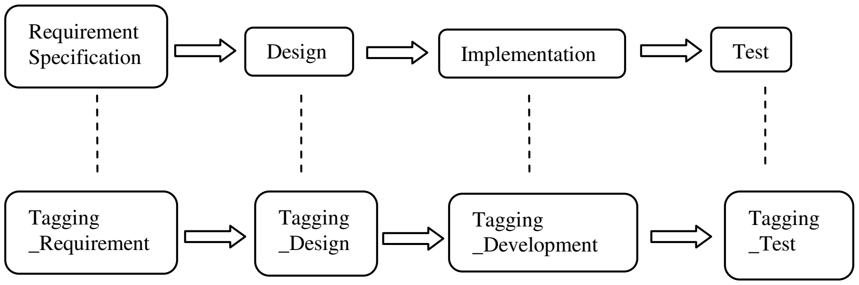

Fig.1. Correlation between Framework and Software Development Lifecycle

It can be seen that the various components of the framework are used during the different phases of software development lifecycle like, the tagging requirement component is used during the requirement specification phase and the tagging development component is used during the implementation phase. The web application interacts with tagging requirement and tagging test components. The developer, on the other hand, interacts with tagging design, tagging development and tagging test components. In the following sub-sections, each of the components is explained.

3.1 Tagging_Requirement

The Tagging_Requirement component is used to elicit requirements of the tagging software. The component consists of six sub-components, namely, Resource Type, Tagging Purpose, Visitor Permissions, Requirement Generator, Requirement Gatherer and Requirement Estimator. As shown in figure 2, type of the resource is identified by web application using Resource Type sub-component. This identification helps in finding out the features of the resource. The Tagging Purpose sub-component allows the web application to identify the purpose of using tags. This process is carried out to determine the kind of restriction is required on the tags. The Visitor Permissions sub-component allows the web application to select the level of flexibility in usage to be provided to the visitor as defined in [2]. The three levels of flexibilities from which the web application can choose are UseTagResource (UsTR), UseTagResource_UpdateTag (UsTR_UpT) and UseTagResource_UpdateTagResource (UsTR_UpTR). The visitor level, UsTR, provides the lowest level of usage among these and the level UsTR_UpTR has the highest level. It is necessary that the visitor will have the lowest level of permission (UsTR) for accessing the system by default. The other two levels of systems are optional and web application can choose either of three available.

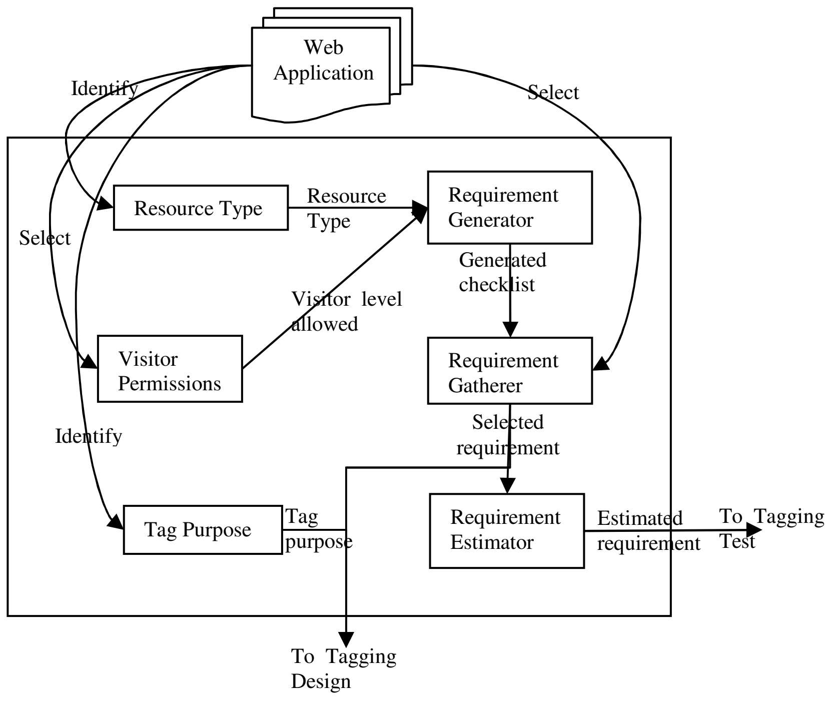

Fig.2. Working of Tagging_Requirement Component

The Requirement Generator sub-component collects the output of two of its peer sub-components, resource type and visitor permissions. It, then, produces the requirement checklist as output based on these two outputs and the weighted requirement checklist presented in [1]. The output of resource type sub-component is used to define the features of the resource in the checklist. The output of visitor permissions is used to determine the features that would be available to the visitor of the tagging software. Below given table 2 shows the scenario applicable, based on the output of visitor permissions sub-component.

Table 2. Influence of Visitor Permission on Tagging Dashboard

Visitor's level			**Tagging Dashboard for Visitor contains**
UsTR	UsTR_UpT	UsTR_UpTR	
Selected	Not Selected	Not Selected	Nothing
Not Selected	*Selected*	Not Selected	Tag Entities only
Not Selected	Not Selected	*Selected*	All Entities

From table 2, it can be seen that, only three cases have been shown because the web application can select only one level of flexibility out of the three. Hence, the other possible cases are not depicted. Also, table 2 depicts that in case, UsTR, lowest level, is selected, then, there is no separate requirement checklist generated. In case, UsTR_UpT is selected, then, only the tag entities of tagging dashboard are included in the checklist. However, if the visitor type, UsTR_UpTR, is selected, then all entities are included in the checklist, since this kind of visitor has maximum privileges in the tagging software.

The Requirement Gatherer sub-component provides checklists to the web application for selection of features. Each checklist has three parts, required-basic, optional-basic and optional-advanced. The web application has to select the required-basic part of checklist, while, the other two parts are optional. The web application can select some, all or no part from these checklists.

The Requirement Estimator sub-component performs the task of quantification of the selected requirements. The sub-component takes as input the output of the sub-component requirement gatherer and conducts the estimation using the weighted requirement checklist estimation formulas described in [1]. Using the formulas, the weighted percentage for each of the checklist is calculated. This calculated value is used afterwards for testing purposes.

3.2 Tagging_Design

The Tagging_Design component is used to create the design of the tagging software that has to be integrated. The tagging design component consists of two sub-components which determine the design of tagging software as displayed in figure 3. The first sub-component is the *Splitter*. This sub-component is assigned the task to split the requirement into the basic buildings blocks of tagging system, namely, resource and tag. The sub-component takes the output of sub-component requirement gatherer as an input divides the selected requirement into resource and tag requirements.

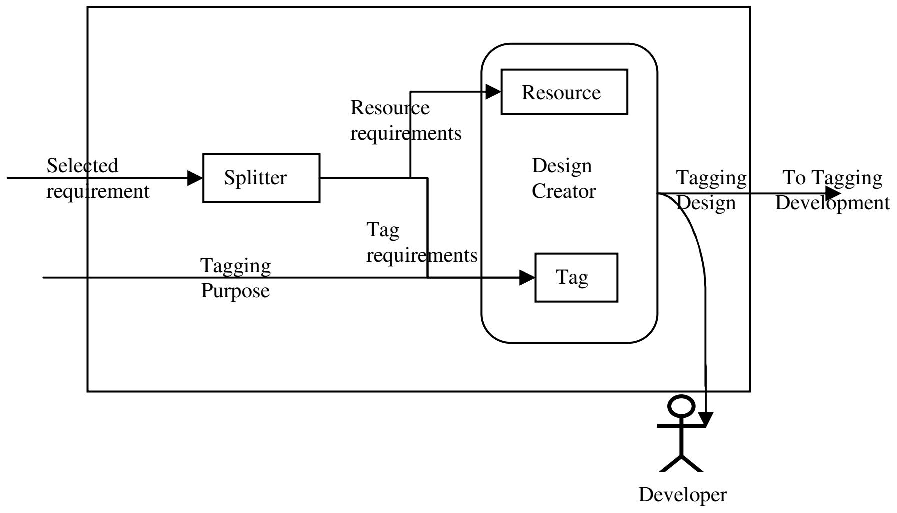

Fig.3. Working of Tagging_Design Component

The second sub-component is *Design Creator*. This sub-component creates the design of the tagging software using the output of its peer sub-component, splitter, and creates the design. The design is based on the structural design presented in [1]. The sub-component consists of two parts, resource and tag. The resource part of the sub-component creates the design for the resource in the tagging system. The tag part takes into consideration the purpose of tagging and incorporates any required restriction in the design for tag.

3.3 Tagging_Development

The Tagging_Development component performs the task of creation of the tagging software as well as integration of tagging software into web application. The component consists of two sub-components, namely, *Creator* and *Integrator* as seen from figure 4. The Creator aids the process of creating tagging software. The Creator and the developer select either of following three options –

- Use some Tagging software
- Use freely available code
- Write new code

The developer can use some already available tagging software like FreeTag [4] or, the developer can use freely available code for the creation of tagging software. However, in both these cases, the developer may need to adapt the tagging software to match the stated requirements as well as design specified. The developer can also use the third option of writing completely new code from start in which case there would be no need for adaptation.

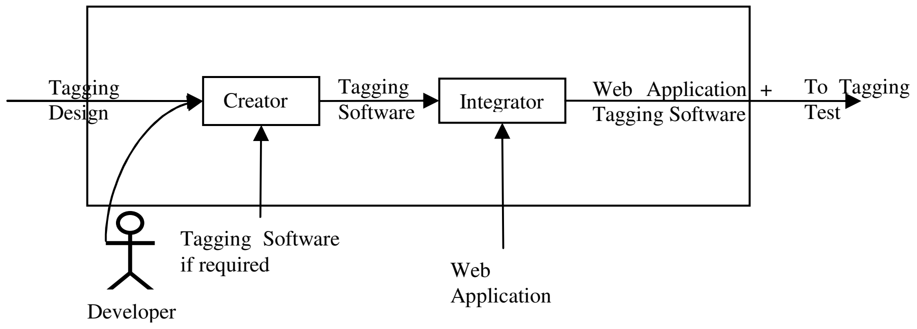

Fig.4. Working of Tagging_Development Component

The second sub-component, Integrator, handles the process of integration of tagging software into web application. The main task of the sub-component is to check whether the created (or adapted) tagging software matches the look and feel of the web application. The developer will have to alter the tagging software according to the changes, if required. Moreover, the Integrator provides guidance as to where should the tagging software be integrated into the web application. Below given table provides guidelines as to where in the web application, the corresponding part of the structural design should be integrated.

Table3. Guidelines for integration of Tagging Software

Part of Structural Design	User Type	Specific Instruction
Resource	Visitor (UsTR)	Public Domain
	Visitor (UsTR_UpTR)	Login-based Mechanism
	Administrator	Private Domain
Tag	Visitor (UsTR)	Public Domain
	Visitor (UsTR_UpT)	Login-based Mechanism
	Administrator	Private Domain

It can be seen that in the Resource part, the entities accessible through Visitor (UsTR) are to be placed in Public Domain and is accessible to any visitor of the site. On the other hand, entities available to Visitor (UsTR_UpTR) should be placed under Login-based Mechanism with access available to visitors only after successful login. Similar premise is followed for the Tag part of the structural design.

3.4 Tagging_Test

The fourth component conducts the task of testing the integrated tagging software. The component contains two sub-components – Verification and Validation as depicted in Figure 5. The Verification sub-component refills the weighted requirement checklists in sub-component requirement gatherer in accordance to the integrated tagging software. Stated simply, those entities and their sub-parts are marked that are present in the integrated tagging software. This is followed by estimation of the weighted percentage for each checklist using the requirement estimation sub-component. The calculated value of this process is, then, compared with the values calculated previously. Ideally, there should be no discrepancy in these two sets of values, with both being same. In case, they are not same, it can be concluded that the tagging software is not built right".

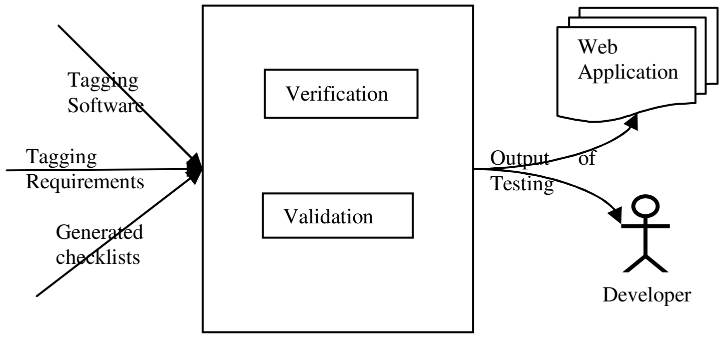

Fig.5. Working of Tagging_Test Component

The Validation sub-component performs task of validating the integrated tagging software. This sub-component takes the refilled weighted requirement checklist from its peer sub-component, Verification, and along with the original weighted requirement checklists, are presented to the web application owner, so that, it can be verified that the tagging software "built is the right thing".

The framework explained in this section expedites integration of tagging software into web application. In the next section, the limitation of the framework as well as the future work is explained.

4. LIMITATION AND FUTURE WORK

The framework presented here is for the purpose of integrating tagging software into web application. However, problem arises if the framework or structural design or the weighted requirement checklists are used for creating a standalone tagging software application. Our framework would not be sufficient for such purpose as standalone tagging software would require much more functionality and features, which are out of scope of this paper.

Also, structural design and weighted requirement checklists are based on study performed of most popular and freely available tagging software. The commercial versions of the tagging software may contain some functionality which is not included in the freely available software. Also, other freely available tagging software may contain some extra features that are not included here. This is a limitation in our work. However, the structural design and the weighted requirement checklists and the derived framework are extensible in nature. They can be easily updated to include any new functionality or feature.

We are in the process of developing a tool based on the framework presented in this paper. The tool would help selecting the requirements as well as the outlining the design of tagging software. Also, the tool would help in testing of the integrated tagging software.

5. RELATED WORK

Much academic work has been carried with respect to tagging. Different fields like, identification of type of tags, behavior of users as well as categories of users has been researched by academicians. Also, several research publications exist for the effect of poorly managed tags in the tagging software. Moreover, a few frameworks have been created for assisting in Chinese Word Segmenting, assessing navigability of tagging system, semantic relation extraction.

Golder and Huberman [5] present information dynamics in "collaborative tagging systems". They standardize the form a tag takes in tagging software. They have defined seven categories of the tag, from which tag can take any form. Robu [6] focus on categorization of tags. Here, delicious.com [7] is used for examining dynamics of collaborative tagging. The aim is to find a categorization scheme that emerges from unsupervised tagging by individual users.

Behavior of users is discussed in Santos-Neto et al. [8]. The authors characterize interest sharing in the system using pair wise similarity between users' activity. They show that interest sharing leads to an implicit structure that exhibit a natural segmentation. Korner et al. [9] distinguishes the users of the tagging software into two parts - categorizers, who categorize resources, and, describers, who describe resources using tags. Schöfegger et al. [10] use supervised learning mechanisms to analyze online tagged academic literature and extract user characteristics from tagging behavior. They conclude that tags convey rich information about their users, which can be helpful for better understanding and for supporting users.

Marvasti et al. [11] use *delicious.com* to find effects of poorly managed tags. According to authors, poorly managed tags obscure much of the collective sense making and implicit community structure. They make suggestions for improving collaborative tagging systems. Helic et al. [12] apply a pragmatic framework to folksonomies for their evaluation. A decentralized search on a network of tags is carried out. Their aim is to provide improved navigability of social tagging systems and to evaluate different folksonomy algorithms from a pragmatic perspective. Tourné et al. [13] performed an empirical study on value of tags in resource classification. They illustrate effects of applying several filtering and pre-processing operations to reduce ambiguity and noise in tags. The results are analyzed to find the increase in quality of resource classification. Zhang et al. [14] examine temporal factor in users' tagging behaviors. The examination is done by investigating occurrence patterns of tags and incorporating this into a novel method for ranking tags.

Zhao et al. [15] creates a framework for character-based Tagging Framework for Chinese Word Segmentation. The authors consider Chinese word segmentation as a character-based tagging problem and provide a framework for solving this problem. Trattner et al. [16] present a framework, NAVTAG, for assessing and improving the navigability of tagging systems. The framework calculates the navigability of tag network using different tag cloud and resource list generation algorithm. Shen et al. [17] propose a framework, called REACTOR, for evaluating real-world enterprise data set and extracting relation extraction from enterprise data.

Generally, a freely available tagging functionality like FreeTag or cocoa [18] is adapted and incorporated into a web application. However, the web application has little knowledge about possible features that tagging software can provide, and their need in a tagging software. Similarly, with absence of design document, the task of updating the tagging software becomes cumbersome for the developer. Thus, some kind of formal specification is required for easing the integration of tagging software into the web application. An extensive search for research papers related of such kind for tagging software in web application has not yielded results.

6. CONCLUSION

A framework for integration of tagging software into web application is presented in this paper. The framework facilitates the specifying of requirements during the software requirement phase. The framework also helps during the design phase by outlining the design of the tagging software. Moreover, the framework is used during the testing phase for verification and validation of the integrated tagging software. The structural design and weighted requirement checklists of the framework can be easily updated to add new features and functionality.

REFERENCES

[1] Gupta, K., Goel, A.: Requirement Estimation and Design of Tagging Software in Web Application. Manuscript submitted for publication (2012).

[2] Gupta, K., Goel, A.: Tagging requirements for web application. In: 5th India Software Engineering Conference (ISEC 2012), pp. 81-90. ACM Digital Library, ACM, Kanpur, India (2012)

[3] Smith, G.: Tagging: people-powered metadata for the social web. New Riders Publishing (2007)

[4] Freetag/freetag - GitHub, https://github.com/freetag/freetag.

[5] Golder, S.A., Huberman, B.A.: The Structure of Collaborative Tagging Systems. Computing Research Repository abs/cs/050. (2005)

[6] Robu, V., Halpin, H., Shepherd, H.: Emergence of consensus and shared vocabularies in collaborative tagging systems. ACM Transactions on the Web 3, 1-34. (2009).

[7] Delicious, http://www.delicious.com/

[8] Santos-Neto, E., Condon, D., Andrade, N., Iamnitchi, A., Ripeanu, M.: Individual and social behavior in tagging systems. In: 20th ACM conference on Hypertext and hypermedia, pp. 183-192. ACM, Torino, Italy (2009).

[9] Korner, C., Kern, R., Grahsl, H.-P., Strohmaier, M.: Of categorizers and describers: an evaluation of quantitative measures for tagging motivation. In: 21st ACM conference on Hypertext and hypermedia, pp. 157-166. ACM, Toronto, Ontario, Canada (2010).

[10] Sch, #246, fegger, K., Korner, C., Singer, P., Granitzer, M.: Learning user characteristics from social tagging behavior. In: 23rd ACM conference on Hypertext and social media, pp. 207-212. ACM, Milwaukee, Wisconsin, USA (2012).

[11] Marvasti, A.F., Skillicorn, D.B.: Structures in collaborative tagging: an empirical analysis. In: Thirty-Third Australasian Conference on Computer Science, pp. 109-116. Australian Computer Society, Inc., Brisbane, Australia (2010).

[12] Helic, D., Strohmaier, M., Trattner, C., Muhr, M., Lerman, K.: Pragmatic evaluation of folksonomies. In: 20th International conference on World Wide Web, pp. 417-426. ACM, Hyderabad, India (2011).

[13] Nicol, #225, Tourn, s., #233, Godoy, D.: Evaluating tag filtering techniques for web resource classification in folksonomies. Expert Syst. Appl. 39, 9723-9729. (2012).

[14] Zhang, L., Tang, J., Zhang, M.: Integrating temporal usage pattern into personalized tag prediction. In: 14th Asia-Pacific International conference on Web Technologies and Applications, pp. 354-365. Springer-Verlag, Kunming, China (2012).

[15] Zhao, H., Huang, C.-N., Li, M., Lu, B.-L.: A Unified Character-Based Tagging Framework for Chinese Word Segmentation. 9, 1-32. (2010).

[16] Trattner, C.: NAVTAG: a network-theoretic framework to assess and improve the navigability of tagging systems. In: 11th International conference on Web engineering, pp. 415-418. Springer-Verlag, Paphos, Cyprus (2011).

[17] Shen, W., Wang, J., Luo, P., Wang, M., Yao, C.: REACTOR: a framework for semantic relation extraction and tagging over enterprise data. In: 20th International conference companion on World Wide Web, pp. 121-122. ACM, Hyderabad, India (2011).

[18] Cocoa Tagging Framework | nudge:nudge, http://www.nudgenudge.eu/taggingframework