

SOFTWARE ENGINEERING FOR TAGGING SOFTWARE

Karan Gupta¹, Anita Goel²

¹Department of Computer Science, University of Delhi, Delhi, India
guptkaran@gmail.com

²Department of Computer Science, Dyal Singh College, University of Delhi, New Delhi, India
agoel@dsc.du.ac.in

ABSTRACT

Tagging is integrated into 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 has been created for integration of tagging software in a web application. The framework follows the software development life cycle paradigms and is to be used during its different stages. 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 popular ones. 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

Software Engineering, 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 software or write a completely new code for tagging. Generally, free tagging software 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 would be used during the Software

Development Lifecycle of the web application. The framework aids the user during requirement elicitation and specification phase. 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 design of tagging software and weighted requirement checklist for easing integration of tagging software. The framework consist 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 is based on software engineering paradigms and aids 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 work associated with our framework. Section 3 describes the related work. The framework is discussed in section 4. Section 5 lists the benefits. Limitations and future work is explained in section 6. Section 7 states the conclusion.

2. BACKGROUND

Web applications use tagging for promoting 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 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 design.

2.1 Actors of Tagging Software

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

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

Of the three actors 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 Checklists

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 during its inclusion in a web application.

Weight – Features of tagging software have different level of popularity. Features like add tag to a resource, delete tags from resource are highly popular. On the other hand, features like describing a tag 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 –

- *Highly-Popular* for specifying highest level of popularity and is denoted by weight ‘3’.
- *Intermediately-Popular* for those that may be helpful in the software but are not a necessary requirement. It is denoted by weight ‘2’.
- *Least-Popular* specifies the lowest level of popularity, and is 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
Resource Use	3	Sharing	3	Share	3
				(System), (User), (Related)	2
Tag Cloud	1	View	1	(Resource), Order(Alphabet/Count), Order(Cloud/List), On(Bundled/ Unbundled/All), Usage(1/2/5), Show/Hide Count	1
				Share	1
Tag Sharing	1	Sharing	1	Share	1

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

2.3 Design for Tagging Software

The two building blocks of tagging software are - Resource and Tags [3]. The 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 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 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 design is accessed by 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 design using two level of permissions - lowest access permission (*UsTR*) and highest access permission (*UsTR_UpTR*). The tag part of 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*.

3. 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. 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.

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. In the next section, we discuss our framework.

4. FRAMEWORK

Web Applications use tagging software to manage large amount of stored information. Generally, the tagging software is integrated into web application on the fly with no formal document or specification available for help. Based on the discussion in the previous section, a framework has been developed to aid the integration of tagging software into web application. The framework is based on software engineering paradigms and is designed to be used during different phases of 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 mapping of the software development lifecycle of a web application and our framework is depicted in Figure 1.

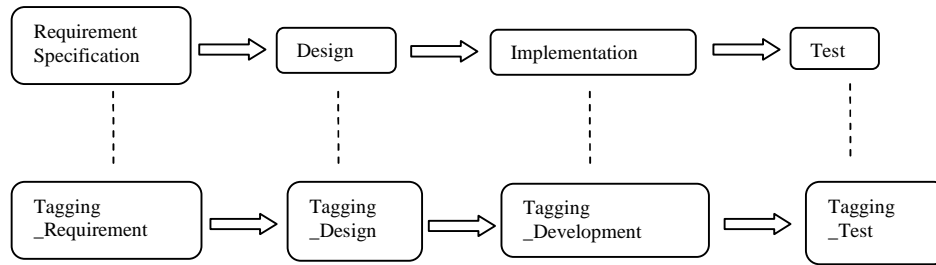


Figure 1. Correlation between Framework and Software Development Lifecycle

It can be seen that the various components of 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.

4.1 Tagging_Requirement

The Tagging_Requirement component is used to elicit requirements of the tagging software. The component consists of five sub-components, namely, *Resource Type*, *Tag Purpose*, *Visitor Permissions*, *Requirement Gatherer* and *Software Estimator*. The component is depicted in figure 2.

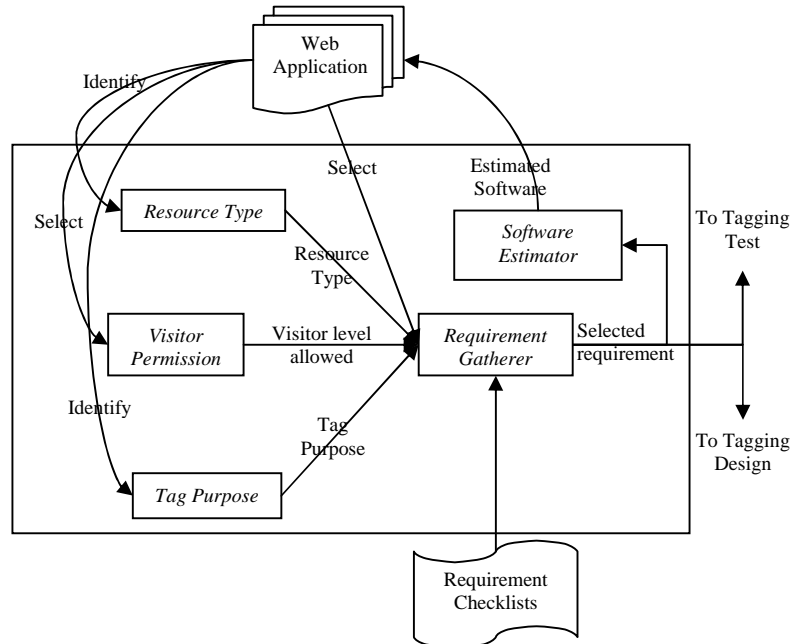


Figure 2. Working of Tagging_Requirement

As shown in figure 2, the *Resource Type* sub-component allows the web application to identify the kind of resource on which tagging has to be performed. This identification helps in finding out the features of the resource. The *Tag 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 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 levels.

The *Requirement Gatherer* sub-component collects the output of three of its peer sub-components, *Resource Type*, *Visitor Permissions* and *Tag Purpose*. It, then, produces the requirement checklist as output based on the output of these three sub-components 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 *Tag Purpose* sub-component aides in determination of restrictions associated to tag. 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 Permissions* on Tagging Dashboard

Visitor's level			Tagging Dashboard for Visitor contains
UsTR	UsTR_UpT	UsTR_UpTR	
<i>Selected</i>	Not Selected	Not Selected	Nothing
Not Selected	<i>Selected</i>	Not Selected	Tag Entities only
Not Selected	Not Selected	<i>Selected</i>	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, then, provides checklists to the web application for selection of features. Each checklist has three parts, *Highly-Popular*, *Intermediately-Popular*, and *Least-Popular*. The web application can select some, all or no part from these checklists. The requirement selected by the web application is, then, provided to the Tagging_Design and Tagging_Test Component.

The *Software Estimator* sub-component performs the task of quantification of the selected requirements. The sub-component takes 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 provided to web application for software estimation purposes.

4.2 Tagging_Design

The Tagging_Design component is used to create the design of the tagging software for integration. 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* and divides the selected requirement into resource and tag requirements.

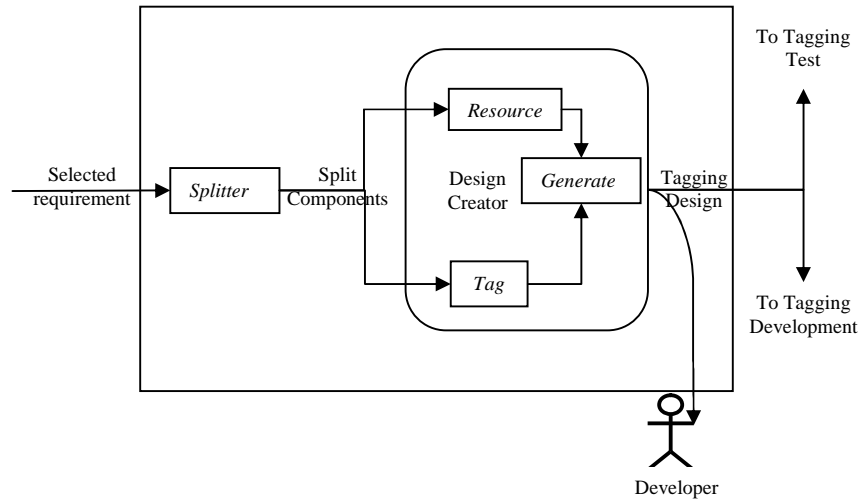


Figure 3. Working of Tagging_Design

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. The design is based on the design of tagging software presented in [1]. The sub-component consists of three parts - resource, tag and generate. The *Resource* part of the sub-component creates the design for the resource in the tagging system. The *Tag* part generates the design for tag in the tagging system. The *Generate* part of the *Design Creator* sub-component, then, uses these generated designs to create the design for the complete tagging software. The generated design of the tagging system is, then, provided to the developer and the Tagging_Development and the Tagging_Test component.

4.3 Tagging_Development

The Tagging_Development component performs the task of creation and integration of tagging software into web application. The component consists of three sub-components, namely, *Code*, *Wrapper* and *Integrator* as seen from figure 4. The *Code* sub-component aids the process of creating tagging software. The developer is provided with two options, (1) select some Tagging APIs or (2) write new code.

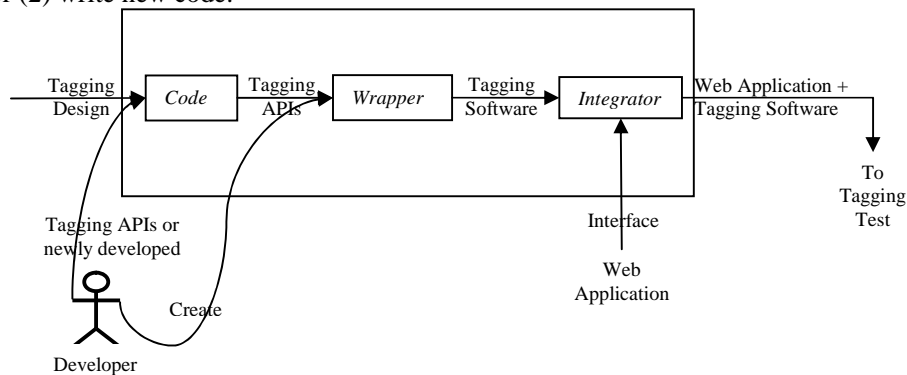


Figure 4. Working of Tagging_Development

The developer can use some already available tagging API like FreeTag [4] or, the developer can exercise the option of writing new code for the tagging software. The second sub-component, *Wrapper*, handles the creation of interface for allowing the use of tagging software into web application. The interface allows the adaptation of the tagging API to match the stated requirements as well as design specified of the web application. In case, the developer has exercised the option of creating new code, the developer may not be required to create a wrapper, if the newly developed code for tagging is in the form of software.

Integrator handles the process of integration of tagging software into web application. The main task of the *Integrator* 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 in regard to integration of tagging software into the web application as depicted in below given table.

Table 3. Guidelines for integration of Tagging Software

Part of 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 design.

4.4 Tagging_Test

The fourth component conducts the task of testing the integrated tagging software. The component consists of *Verification & Validation* sub-component as depicted in Figure 5. The Verification & Validation sub-component uses the selected requirements, the tagging design and the integrated tagging software to test whether the integration has carried out successfully or not. The *Verification & Validation* sub-component verifies the statements that “the tagging software is built right” and “the tagging software built is the right thing”.

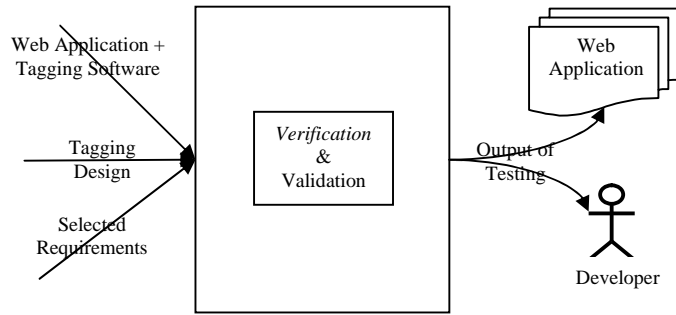


Figure 5. Working of Tagging_Test

The framework explained in this section expedites integration of tagging software into web application. In the next section, the benefits of using the framework are explained.

5. BENEFITS OF FRAMEWORK

Both the developer and the owner of web application can use the framework presented here during the integration of tagging software into web application. The framework helps the developer in understanding the structure of tagging software. Also, the design explains the interaction that occurs between the tagging software and its users to the developer. Since the framework is mapped into the SDLC of the web application, the developer can use the framework during different stages of SDLC to accomplish various tasks. For example, during the design stage, the developer can use the framework to create design while during the testing stage, the developer uses the framework for testing the integrated tagging software.

The web application is also benefited by the framework. The web application uses the framework during the requirement to select the features of tagging software. The framework, then, provides the web application, an estimate of the software selected. Also, during the testing phase, the web application can use the framework for testing the integrated tagging software. Moreover, the extensibility of the design and the weighted requirement checklists allows the framework to be extensible in nature and allows any feature or functionality.

6. LIMITATION AND FUTURE WORK

The framework presented here is for the purpose of integration of tagging software into web application. Moreover, the associated design or the weighted requirement checklists are used for the purpose of integration of tagging software into web application. However, the framework or the associated design or weighted requirement checklist may not be used for creating a standalone tagging software application. Our framework may 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, 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. This is a limitation in our work. However, the 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.

7. CONCLUSION

A framework for integration of tagging software into web application is presented in this paper. The framework is based on software engineering paradigms. 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 design and weighted requirement checklists of the framework can be easily updated to add new features and functionality.

REFERENCES

- [1] Gupta, Karan and Goel, Anita, (2012), "Requirement Estimation and Design of Tagging Software in Web Application". Manuscript submitted for publication (2012).
- [2] Gupta, Karan and Goel, Anita, (2012), "Tagging requirements for web application", Proceedings of the 5th India Software Engineering Conference, ACM, pp. 81-90.
- [3] Smith, Gene, (2007), Tagging: people-powered metadata for the social web, New Riders Publishing, p. 216.
- [4] Luk, Gordon, "freetag • GitHub", <https://github.com/freetag>
- [5] Golder, Scott A. and Huberman, Bernardo A., (2005), "The Structure of Collaborative Tagging Systems", Computing Research Repository.
- [6] Robu, Valentin, et al., (2009), "Emergence of consensus and shared vocabularies in collaborative tagging systems", ACM Transactions on the Web (TWEB), vol. 3, no. 4, pp. 1-34.
- [7] Hurley, Chad and Chen, Steve, "Delicious", <http://www.delicious.com/>.
- [8] Santos-Neto, Elizeu, et al., (2009), "Individual and social behavior in tagging systems", Proceedings of the 20th ACM conference on Hypertext and hypermedia, ACM, pp. 183-192.
- [9] Körner, Christian, et al., (2010), "Of categorizers and describers: an evaluation of quantitative measures for tagging motivation", Proceedings of the 21st ACM conference on Hypertext and hypermedia, ACM, pp. 157-166.
- [10] Schöfegger, Karin, et al., (2012), "Learning user characteristics from social tagging behavior", Proceedings of the 23rd ACM conference on Hypertext and social media, ACM, pp. 207-212.
- [11] Marvasti, A. Fani and Skillicorn, D. B., (2010), "Structures in collaborative tagging: an empirical analysis", Proceedings of the Thirty-Third Australasian Conferenc on Computer Science - Volume 102, Australian Computer Society, Inc., pp. 109-116.
- [12] Helic, Denis, et al., (2011), "Pragmatic evaluation of folksonomies", Proceedings of the 20th international conference on World Wide Web, ACM, pp. 417-426.
- [13] Tournéa, Nicolás and Godoy, Daniela, (2012), "Evaluating tag filtering techniques for web resource classification in folksonomies", Expert Systems with Applications, vol. 39, no. 10, pp. 9723-9729.
- [14] Zhang, Lei, et al., (2012), "Integrating temporal usage pattern into personalized tag prediction", Proceedings of the 14th Asia-Pacific international conference on Web Technologies and Applications, Springer-Verlag, pp. 354-365.
- [15] Zhao, Hai, et al., (2010), "A Unified Character-Based Tagging Framework for Chinese Word Segmentation", vol. 9, no. 2, pp. 1-32.
- [16] Trattner, Christoph, (2011), "NAVTAG: a network-theoretic framework to assess and improve the navigability of tagging systems", Proceedings of the 11th international conference on Web engineering, Springer-Verlag, pp. 415-418.
- [17] Shen, Wei, et al., (2011), "REACTOR: a framework for semantic relation extraction and tagging over enterprise data", Proceedings of the 20th international conference companion on World wide web, ACM, pp. 121-122.
- [18] Hoffart, J and GbR, DB, "Cocoa Tagging Framework," 2006; <http://www.nudgenudge.eu/taggingframework>.