Automated Support for Combinational Creativity in Requirements Engineering

Tanmay Bhowmik*, Nan Niu†, Anas Mahmoud*, and Juha Savolainen‡
* Department of Computer Science and Engineering, Mississippi State University, USA
† Department of Electrical Engineering and Computing Systems, University of Cincinnati, USA
‡ Danfoss Power Electronics A/S, Graasten, Denmark
tb394@msstate.edu, nan.niu@uc.edu, amm560@msstate.edu, JuhaErik.Savolainen@danfoss.com

***Abstract*—Requirements engineering (RE), framed as a creative problem solving process, plays a key role in innovating more useful and novel requirements and improving a software system's sustainability. Existing approaches, such as creativity workshops and feature mining from web services, facilitate creativity by exploring a search space of partial and complete possibilities of requirements. To further advance the literature, we support creativity from a combinational perspective, i.e., making unfamiliar connections between familiar possibilities of requirements. In particular, we propose a novel framework that extracts familiar ideas from the requirements and stakeholders' comments using topic modeling and applies part-of-speech tagging to obtain unfamiliar idea combinations. We apply our framework on two large open source software systems and further report a human subject evaluation. The results show that our framework complements existing approaches by generating original and relevant requirements in an automated manner.**

***Index Terms*—Requirements engineering; creativity; topic modeling; requirements elicitation**

I. Introduction

Much of traditional requirements engineering (RE) has considered that requirements exist in the stakeholders' minds in an implicit manner [1], and has focused on models and techniques to aid identification and documentation of such requirements. Modern software industry, however, has become extremely competitive as we find multiple software products striving to serve the users in the same application domain. In order to sustain, a software system needs to distinguish itself from other similar products and consistently enchant customers with novel and useful features. As a result, requirements engineers need to create innovative requirements in order to equip the software with competitive advantage. To that end, RE, framed as a creative problem solving process, plays a key role in innovating more useful and novel requirements, thereby improving a software system's sustainability [2], [3].

Creativity, a multidisciplinary research field, can be considered as "the ability to produce work that is both novel (i.e., original and unexpected) and appropriate (i.e., useful and adaptive to task constraints)" [4]. According to Maiden *et al.* [2], creativity in RE is the capture of requirements that are new to the project stakeholders but may not be historically new to humankind. It has been suggested that stakeholders may obtain creative requirements by exploring, combining, and transforming existing ideas in the conceptual domain [2], [5]. Note that creativity may be more related to novelty, while innovation also requires some demonstrated value or utility. In this sense, our current work focused more on creativity.

In order to aid creativity in RE, recent research has investigated several approaches. Maiden and colleagues [3], [6], [7], [8] conducted creativity workshops on exploring technical and psychological aspects of creativity and suggested integrating these aspects in the RE process. Techniques, such as generating requirements with scenario, have been proposed to support creativity while exploring information analogical to the current context [9], [10]. In a recent study, Hariri *et al.* [11] presented a framework to obtain requirements by mining feature descriptions of similar products from online product listings. These contemporary approaches facilitate creativity by *exploring* a search space of partial and complete possibilities of requirements. To further advance the literature, we support creativity from a *combinational* perspective, i.e., making unfamiliar connections between familiar possibilities of requirements [5], [12].

In this paper, we propose a novel framework that mines ideas familiar to the stakeholders and creates new requirements by obtaining unfamiliar connections. It has been suggested that people belonging to the same social group are generally interested in similar ideas and share common knowledge [13], [14]. Accordingly, in order to extract familiar ideas, we mine the requirements commonly discussed by distinct stakeholder groups. To that end, we first group the stakeholders by clustering the network created based on stakeholders' social interaction. Then, we obtain ideas in terms of dominant topics [15] by applying Latent Dirichlet Allocation (LDA), the most commonly used technique for topic modeling in natural language processing [16]. We further achieve unfamiliar combinations of the dominant ideas by exploiting part-of-speech (POS) tagging [17]. We apply our framework on Firefox[1] and Mylyn [18], two large open source software (OSS) systems. We further conduct a human subject evaluation and the results indicate promising practical implications of our framework.

The contributions of our work lie in an advancement of the current solutions that facilitate creativity practice in RE. Our framework provides automated support for combinational creativity and complements existing approaches by generating original and relevant requirements. The rest of the paper is

[1] http://www.mozilla.org/en-US/firefox/new/

organized as follows. Section II covers background information and related work. Section III introduces our framework. Section IV describes the creation of new requirements for our subject systems. Section V details the human subject evaluation followed by Section VI presenting further discussion and the limitations of the work. Finally, Section VII concludes the paper with an outline of our future work.

II. Background and Related Work

A. Creativity in RE

Creative ideas: Being novel and being appropriate are the two intrinsic attributes of an idea to be creative [4]. An idea can be novel from three different aspects: H-Creativity — new to a person-kind (i.e., historically creative) [5], P-Creativity — new to a person but not to the person-kind or others (i.e., psychologically creative) [5], and S-Creativity — idea for a specific task which is novel in the particular situation or domain (also known as situated creativity) [19]. Meanwhile, an idea is appropriate if it is useful to accomplish a task and can be adapted following the task constraints [4]. According to Maher *et al.* [20], from a design perspective, an idea can be creative if it instigates surprise in terms of deviation in patterns of outcomes. Maiden and colleagues [2], however, suggest creativity in RE to be mostly situated creativity, i.e., creating requirements and other outcomes new to project stakeholders but need not be historically new.

Over the last decade, several techniques have been proposed in order to measure the novelty of a new requirement. Ritchie [21] posited a set of formal criteria that could be applied to assess the creative behavior of software programs. Measuring dissimilarity to existing domain examples could be a way of determining novelty of a requirement [12]. In order to invent requirements from software, Zachos and Maiden [10] exploited requirements similarity matching engines and judged novelty by computing dissimilarities among analogical matches. In the creativity framework proposed in this paper, we exploit the idea of measuring dissimilarity in finding unfamiliar idea combinations.

Categories of creativity: Following Boden [5], creativity in RE is categorized into three groups depending on the techniques and heuristics used [12]. 1) In *exploratory creativity*, creative requirements are obtained by exploring a partial and complete possibilities in the search space. This exploration is guided by rules and task constraints specific to the intended software system. 2) *Combinational creativity* is achieved by making unfamiliar connections between known requirements in a familiar setting. 3) The third way of accomplishing creativity in RE is to challenge the constraints on the search space and to enlarge the space of possible requirements to be explored. Creativity attained by this means is known as *transformational creativity*.

Figure 1 presents a conceptual picture of the three categories of creativity. Let us assume a creativity scenario for a hypothetical software product S: "provide access control" is a current requirement and limitation on available hardware is an initial constraint. Let XYZ be a search space with possible

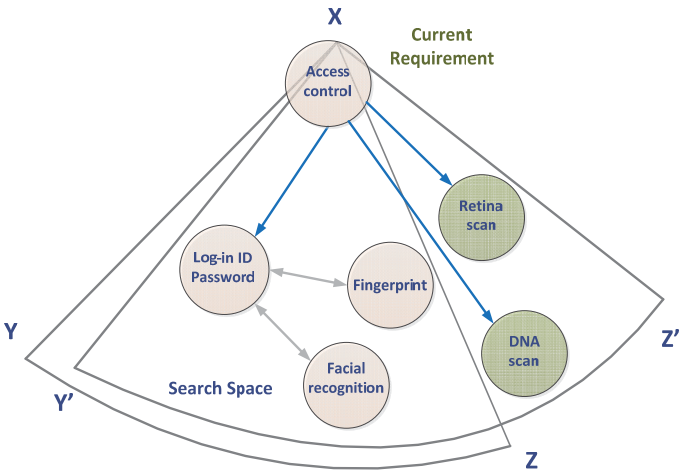


Fig. 1. Categories of creativity based on techniques and heuristics (adapted from [12]).

requirements "log-in ID and password", "finger print", and "facial recognition". Provided that they satisfy the system constraints, using any of these options for access control is an instance of exploratory creativity. Combination of two apparently different access control means, such as log-in ID and password along with finger print, or log-in ID and password combined with facial recognition, can be considered as combinational creativety. Now, let us further consider that the initial constraint on hardware limitation is relaxed and we enlarge the search space towards the biometric direction, thereby obtaining the new search space $XY'Z'$. Options, such as DNA and retina scan, could also be available due to this expansion, an instance of transformational creativity.

On the way to creative requirements: Research has conducted creativity workshops and presented several frameworks in order to incorporate creativity techniques and heuristics in a direct or indirect manner. Zachos and Maiden [10] conducted creativity workshops in order to identify requirements for the Fiat real-time parking space booking system. They performed an analogical mapping between hotel reservation and parking space booking, and explored online hotel reservation systems to create requirements. In order to discover features for a future air space management software system, exploratory creativity was followed during the analysis of requirements and emergent system properties [3]. Lutz and colleagues [22] presented an approach that performed KAOS Obstacle Analysis to explore requirements in a space defined by obstacles for a safety-critical, autonomous system. Salinesi *et al.* [23] proposed a prototype tool that performed requirements-based product configurations within constraints. This tool discovered various permitted features for a new product in a product line. The i^*/TROPOS proposed by Fuxman *et al.* [24] exploited model checking techniques on the explored space of specification properties in an attempt to avoid unreasonable requirements. All these frameworks and tools mostly incorporate exploratory creativity, directly or indirectly, and are concerned with creating requirements for a new system or a product line.

Lately, the collaborative nature of creativity in RE has been highlighted by Mahaux and colleagues [25]. Their research shows that people often need to work collaboratively to be creative and provides a framework characterizing the collaborative creative process in RE. Following their findings, in this paper, we consider the collaborative attribute of creativity and take into account not only the requirements descriptions but also the comments posted by stakeholders during their collaboration. To that end we utilize the concept of stakeholders' social network in finding their collaboration groups.

B. Stakeholders' Social Network in Software Engineering

Stakeholders and social networks based on their interactions have been widely studied in RE and other software engineering areas, e.g., software maintenance. Damian *et al.* [26] presented the concept of requirements-centric social network by defining social network among stakeholders working on same or interdependent requirements. Begel *et al.* suggested that people could "be friends" by working on the artifacts they share among them [27]. In order to create a visual representation for stakeholders' socio-technical relationship, Sarma *et al.* [28] considered both emails among developers and comments in Bugzilla issue tracking system. Posting and reading comments by stakeholders were also considered by Wolf and colleagues as a means to represent communication flow [29]. Building on the prior research, we consider in our work posting comments and artifacts on issue tracking systems as social interaction among stakeholders.

C. Topic Modeling with Latent Dirichlet Allocation (LDA)

LDA was first introduced by Blei *et al.* [16] as a statistical model for automatically discovering topics in large corpora of text documents. The main assumption is that documents in a collection are generated using a mixture of latent topics, where a topic is a dominant theme that describes the concept of the corpus's subject matter. LDA's scalability, language-independency, as well as its ability to work with incomplete text have made it an appealing analysis model for several software engineering activities [30], [31], [32]. Because the requirements of a software system as well as stakeholders' comments typically contain texual descriptions, LDA becomes particularly useful for our framework. Such textual content can be analyzed to produce latent topic structures for the requirements where every requirements description, associated with stakeholder comments, is analogous to an individual document.

Mathematically, a topic model can be described as a hierarchical Bayesian model that associates a document d in a document collection D with a probability distribution over a number of topics T. In particular, each document d in the collection $(d_i \in D)$ is modeled as a finite mixture over T drawn from a Dirichlet distribution with parameter α, such that each d is associated with each $(t_i \in T)$ by a probability distribution of θ_i. On the other hand, each topic t in the identified latent topics $(t_i \in T)$ is modeled as a multidimensional probability distribution, drawn from a Dirichlet distribution β, over the set of unique words in the corpus (W), where the likelihood of a word from the corpus $(w_i \in W)$ to be assigned to a certain topic t is given by the parameter ϕ_i.

LDA takes the documents collection D, the number of topics K, and α and β as inputs. Each document in the corpus is represented as a bag of words $d =< w_1, w_2, \ldots, w_n >$. Since these words are observed data, Bayesian probability can be used to invert the generative model and automatically learn ϕ values for each topic t_i, and θ values for each document d_i. In particular, using algorithms such as Gibbs sampling [33], an LDA model can be extracted. This model contains, for each t, the matrix $\phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$, representing the distribution of t over the set of words $< w_1, w_2, \ldots, w_n >$, and for each document d, the matrix $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$, representing the distribution of d over the set of topics $< t_1, t_2, \ldots, t_n >$. The topic with the highest probability of occurrence in d is the most dominant topic for d. Therefore, for the document collection D, the topic that becomes dominant the greatest number of times is the most dominant topic for D.

Topic modeling in software engineering: Topic modeling has recently been used in several research areas of software engineering, such as mining software repositories (MSR) [15], [32], [34], requirements traceability [30], and software evolution [35]. Linstead *et al.* [15] applied LDA topic modeling technique on the source code of different versions in order to analyze software evolution. Linstead and colleagues [34] further used topic modeling on Internet-scale software repositories, and summarized program function and developer activities by extracting topic-word and author-topic distributions. The use of topic modeling over source code has been validated and it has been found that the evolution of source code topics is indeed caused by actual change activities in the code [32]. Asuncion *et al.* [30] proposed an automated technique that combined traceability with topic modeling and performed semantic categorization of artifacts during the software development process.

The above efforts follow a common approach in that they apply topic modeling on source code written in computer programming languages. In the creativity framework presented in this paper, one of the objectives is to extract existing ideas from documents mostly written in a natural language (e.g., English). To that end, we adopt LDA, perhaps the most proven topic modeling technique for NLP, thereby generating the underlying dominant themes from requirements and comments posted by stakeholder groups. In the next section, we present a detailed discussion of our creativity framework.

III. Our Creativity Framework

Figure 2 presents an overview of our framework that applies a combinational creativity technique to obtain new requirements for an existing system. The framework starts with a social network based on stakeholders' social interaction, goes through several phases involving techniques, such as clustering, topic modeling, POS tagging, and similarity analysis, and ultimately creates new requirements in an automated manner. These requirements could be considered as an initial baseline

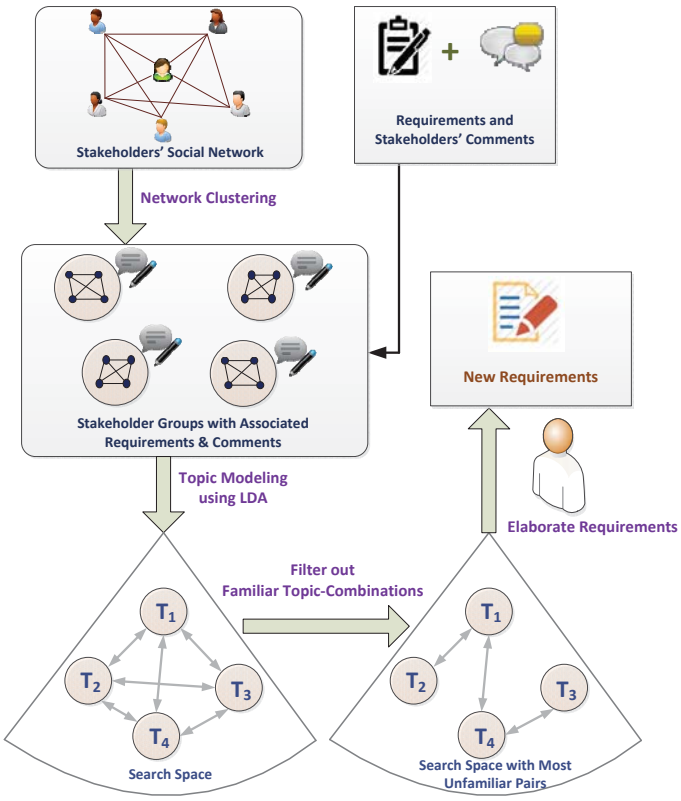


Fig. 2. A framework for combinational creativity.

for creative requirements that can further be discussed among stakeholders for analysis and modification. In the rest of this section, we discuss the phases of this framework in details.

1. Building the social network: The first phase of our framework is to build a weighted connected graph representing the stakeholders' social network. This network should be built based on stakeholders' communication, i.e., two people communicating among themselves should be connected by an edge and how strongly (or frequently) they communicate should be reflected by the edge weight. As several communication means could be followed by stakeholders (cf. Section II-B), our framework does not set any restriction on what activities should be considered as stakeholders' social communication. Depending on the practice followed in a specific software development environment, any set of well defined and properly recorded communication means should be suitable.

2. Clustering the social network: This phase involves clustering the social network in order to obtain stakeholders' social groups. The idea is that the members in the same group have more frequent interaction, whereas there is sparse communication among people belonging to different groups [14]. Our framework is flexible from the perspective of clustering in that any suitable network clustering algorithm [36] can be used as long as necessary information required for the clustering algorithm is available for the social network in concern. Tool supports, both commercial and open source, are available that can be used to perform this clustering activity [37], [38].

3. Extracting familiar ideas: As people belonging to the same social group are generally interested in similar ideas [13], [14], this phase of our framework involves identifying such ideas. In doing so, requirements and comments posted by each member in a social group are collected as text documents, one for each stakeholder. Let us assume that i number of groups have been obtained after the clustering phase. If there are N_i number of members in the i-th group, there will be a collection of N_i documents. LDA is applied on each document collection N_i in order to obtain the topic-word distribution matrix ϕ and document-topic distribution matrix θ. Irrespective of the size of the document collection, LDA always generates t topics where t is a positive natural number (often 100 [15]) chosen by the user. As both ϕ and θ provide the probability distribution of a large number of words and topics respectively, the number of words and topics should be considerably reduced to avoid an explosion of idea combinations in the later phases of this framework. To that end, the following procedure is pursued for each document collection N_i.

- We use the five most probable words from each topic as representatives of the topic's subject; 5 to 3 words were found to be sufficient to convey the topic's subject [39].
- We use the most probable (dominant) topic of each document to represent the document. Formally, a dominant topic can be described as $\theta_{i,j} = max\{\theta_{h,j}, h = 1...k\}$.
- The topics are sorted in descending order based on the number of documents they are dominating.
- These numbers are plotted against the topics and the cutoff is taken based on the trend, thereby obtaining a smaller number of topics for the social group.

4. Obtaining unfamiliar combinations of familiar ideas: Extracted dominant topics provide us a search space of familiar ideas (cf. Figure 2). Our objective is to make unfamiliar connections between familiar possibilities in the search space. To that end, we aim to combine words from two topics, one word from each topic, coming from two different stakeholders' groups. If there are 10 groups with 5 dominant topics per group and 5 words per topic, there will be ${}_{10}C_2 \times {}_{25}C_1 \times {}_{25}C_1 = 28,125$ unique word pairs. This will lead to a combinatorial explosion problem for systems with a large number of diverse stakeholders. In order to tackle this issue, we follow the work on semantic analysis in RE [40], [41] to tease out the action-oriented theme of a requirement. Such theme, according to Fillmore's case theory [42], can be characterized by the *verb* in a requirements description and the *direct object* that the verb acts on. Building upon this knowledge, we structure this phase of the framework with the following two steps, as illustrated in Fig. 3.

- *Flipping the part-of-speech*: For each topic word, we identify its common POS in the existing requirements and comments over the original corpus using a POS tagger. POS tagging is recently being used in text based software engineering tools, such as SWUM [43] and POSSE [44]. We take the most common verb from a topic and the most common noun (object) from another, where two

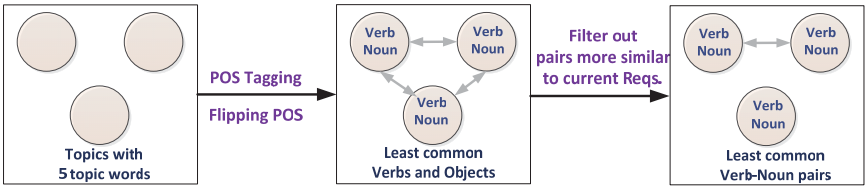


Fig. 3. Finding the least common verb-noun pairs: the leftmost box in the picture contains three topics, represented by three circles, with 5 words each.

topics belong to two separate groups of stakeholders, and consider the words as noun (object) and verb respectively. We identify all such verb-noun pairs.

- *Finding system specific unfamiliar pairs*: To further ensure unfamiliarity, we rank the verb-noun pairs based on their average textual similarities [45] with the current requirements. Then, we filter out the combinations with higher similarity values following a relative filtering approach [45], thereby reducing the search space to most unfamiliar verb-noun pairs (cf. Fig. 2).

5. Elaborating requirements from verb-noun pairs: All the word pairs obtained from the previous phase are presented to a human analyst, preferably a stakeholder proficient in the software's functional attributes. The analyst is also supplied with the semantic and contextual information about the words, as well as more frequently used phrases around the words in the existing requirements. Equipped with these resources, the analyst shall phrase statements following some syntax, e.g., subject + verb + noun/object, and further elaborate the statement using contextual information to obtain new requirements. The subject could be the software system itself (e.g., Firefox) or a suitable phrase from the context. Our framework is flexible in this regard as the analyst is free to elaborate the requirements in her own words as long as the *ideas* provided by the verb-noun pairs are reserved. Further technical details of the framework is presented in Section IV.

IV. Creating Requirements Using Our Framework

This section explains our procedure of examining how the proposed framework supports combinational creativity in RE. In particular, we detail the activities we perform to tease out original, unexpected, useful, and adaptive requirements following the specific phases discussed in Section III.

A. Methodology

In order to test our framework, we select two OSS systems: Firefox and Mylyn [18]. We select these projects as our subject systems for a number of reasons. First, they are large OSS systems and were previously studied in software engineering research [18], [46]. Second, they are very successful applications and can be considered representatives of their own domains. Third, the relevant data about these systems, required to conduct this study, are freely available online over Bugzilla. This enables other researchers to replicate our study. Next is a brief description of our chosen systems.

- *Firefox*: A very successful open source project and a dominating Web browser since its first release in 2004. From November 2004 to June 2011, Mozilla released Firefox stable versions 1.0 through 5.0 and after that made some rapid releases.[2] We collect data about the closed requirements (feature requests) of the stable versions.
- *Mylyn*: A stable plug-in that monitors programmer activity in the Eclipse IDE [18]. It was first started as a part of the PhD thesis supervised by Gail Murphy at the Software Practices Lab at UBC.[3] We consider the closed requirements of Mylyn from its starting in 2005 till February 2012.

For every requirement, we collect information as follows: requirement ID, description, comments, proposer (i.e., stakeholder who proposed the requirement), and stakeholders posting comments and artifacts. All the information, directly available from the requirements page, is collected by running a Web scraping tool written in Java. Table I presents the collected data that we analyze for the subject systems. Note that we observe many requirements marked as duplicates (specially in case of Firefox), and exclude them from our study.

B. Creative Requirements via Idea Combinations

Building the social network: For each subject system, the social network is a weighted graph where each node represents a stakeholder. An edge in the graph represents the communication among two stakeholders and the weight of this edge indicates the total instance of communications between them. To define the weighted edges, we adopt the approach presented by Wolf *et al.* [29]. Let X and Y be two stakeholders and R be a requirement that both X and Y contribute to. We identify an edge XY representing communication between X and Y if: 1) X is the proposer of R or has posted a comment or artifact about R that is read by Y; or 2) Y is the proposer of R or has posted a comment or artifact about R that is read by X. As issue trackers do not keep direct trace of a stakeholder's reading activity, following Wolf *et al.* [29], we assume Y read the information posted by X about R if and only if Y also made a posting. We aggregate such communication instances between X and Y over the analyzed history and obtain the weight of an edge XY.

Obtaining stakeholders' groups: In order to identify stakeholders' groups, i.e., people who interact more frequently among themselves, we cluster the social networks built in the previous phase. To that end, we use Ucinet [37],

[2]http://www.mozilla.org/en-US/firefox/releases/
[3]http://www.eclipse.org/mylyn/about/

TABLE I
DATA COLLECTION OF SUBJECT SYSTEMS

System	Application domain	Analyzed history	# of req.s	Avg. # of comments per req.	# of code files	Written in
Firefox	Web browser	2004–2011	983	18	1,968 (C/C++)	C/C++, JavaScript
Mylyn	Eclipse plug-in	2005–2012	445	11	2,321	Java

TABLE II
CLUSTERING RESULTS

System	# of stakeholders in social network	# of clusters (groups)	Avg. group size*	Fit value
Firefox	783	36	22 ($\pm$8.29)	0.783
Mylyn	136	9	16 ($\pm$6.82)	0.817

*The average value rounded to the next round number.

TABLE III
TOPICS OBTAINED

System	# of clusters	Total # of topics	Avg. # of topics per cluster*	Possible unique word pairs
Firefox	36	318	9 ($\pm$4.15)	1,239,150
Mylyn	9	48	6 ($\pm$3.72)	29,864

*The average value rounded to the next round number.

which provides social network clustering and related features. Ucinet [37] takes the total number of expected clusters k as input and applies hierarchical clustering algorithm based on node similarities. In our context, a higher edge weight means higher similarity between nodes. The output is a text file that elicits the clusters and also provides a fit value where a lower fit value indicates better cluster quality [37]. For both Firefox and Mylyn, we start the clustering process with k=2, observe the fit values by gradually increasing k, and stop further clustering when there is no more reasonable decrease in the fit value. Table II presents the clustering results.

Familiar ideas from stakeholders' groups: Phase 3 of our framework applies LDA [16] on the requirements and comments from all the stakeholders in a social group. For this activity, we use JGibbLDA.[4] This particular implementation uses Gibbs sampling for parameter estimation and inference [47]. From the topic-word matrix and document-topic matrix produced by JGibbLDA, we extract the dominant topics following the heuristics presented in Section III. Along with these matrices, JGibbLDA also produces a topic-words file from which we pick the top 5 words for each topic as topic words. It should be noted that we filter out common key words, such as Firefox and Mylyn, based on the system's context along with frequently used English stop words to avoid noise. However, we find some words appearing in multiple topics, such as Web in case of Firefox and task in case of Mylyn. In such cases, the word is assigned to the topic where it shows the highest probability of occurrence. The results after this phase is summarized in Table III.

The column 'Possible unique word pairs' in Table III presents the number of unique word pairs considering one word per topic from two different stakeholder groups. The high number of possible combinations makes it apparent that without further filtering, elaborating requirements from the word pairs will be very daunting for an analyst. Furthermore, not all word pairs will make much sense so that a meaningful requirement could be generated. The filtering phase that we go through next is specifically designed to tackle this issue.

[4]http://jgibblda.sourceforge.net/

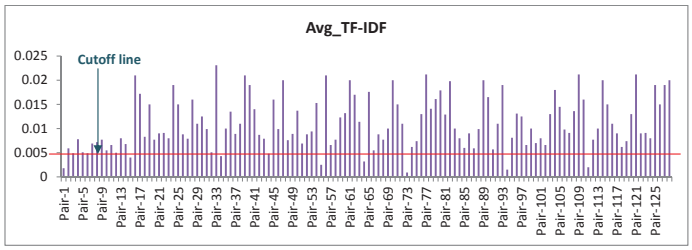


Fig. 4. Finding the least common verb-noun pairs for Mylyn.

Unfamiliar idea combinations for Firefox and Mylyn: This phase first uses Brill's tagger [17] to identify the most common POS for every topic word in the existing requirement descriptions over the original corpus. This allows us to find the most common noun (object) and the most common verb based on the word probabilities in the topic-words file produced in the previous phase. We consider them as least common verb and least common noun respectively, thereby producing the least familiar verb-noun pairs (from the system's perspective) for both Firefox and Mylyn.

Next, following the experience presented in our previous work [48], we calculate TF-IDF cosine similarities between a verb-noun pair and the existing requirements. We average the similarity values obtained for every pair and operationalize a relative filtering scheme to filter out the pairs with higher average similarities. To that end, we extract the verb-noun pairs with $\textit{average similarity} \leq 0.20 * \textit{highest similarity}$ and consider them as the final set of verb-noun pairs. Noted that the level of this cutoff is subject to calibrate for the requirements engineer to work with a manageable set of idea pairs for further expansion. Figure 4 demonstrates the filtering scheme for Mylyn. Table IV summarizes the results after this phase.

Elaborating requirements from pairs: In order to carry out this phase, we recruited a professional software engineer named Bob (pseudonym), working at a local software devel-

TABLE IV
UNFAMILIAR IDEA COMBINATIONS

System	Possible idea combinations		
	Initial	After POS tagging	Final
Firefox	1,239,150	2436	**34**
Mylyn	29,864	128	**9**

opment company. As part of his current job, Bob performs requirements analysis related activities at a regular basis. At work, he executes most of his development activities in Java using Eclipse IDE, and has been using both Firefox and Mylyn for several years. We provided Bob with the final set of word pairs (i.e., unfamiliar idea combinations), automatically generated contextual information for each word, and a set of sample templates to guide the elaboration. We asked Bob to come up with as many new requirements as possible using the word pairs within a time frame of two hours. Bob was requested to preserve the syntax 'verb and noun (object) the verb acts upon' as much as possible, and was allowed to use the Internet, if necessary. A researcher was present to explain the task and the provided artifacts, observe Bob's activities during this elaboration process, and conduct an exit interview.

Figure 5 demonstrates the elaboration of a new requirement for Firefox. The words 'text' and 'number' were obtained from the topics $<support, text, lock, login, enter>$ and $<build, compile, number, tool, item>$ respectively. The dotted boxes contain recommended context information. Note that Bob did not use any recommended word from the topic list for 'text' while writing the requirement "Firefox user can text phone number from the web page." After two hours, Bob elaborated 8 requirements for Firefox and 5 for Mylyn, as shown in Table V. The last column contains the final requirements descriptions after *refinement* and some nouns and verbs do not preserve their initially assigned POS anymore. Section VI provides further discussion on this issue. Next we present a human subject evaluation of our framework.

V. HUMAN SUBJECT EVALUATION

A. Study Setup

We recruited 29 developers with experience in Java and C#, including both undergraduate and graduate students and staff programmers from our institute. The developers participated voluntarily by responding to an email invitation. We made a confidentiality agreement with the participants to respect their anonymity. Each participant worked individually in a lab and began by signing the consent form. The demographic

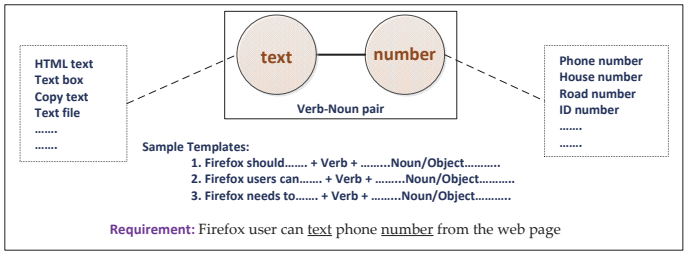


Fig. 5. Example for requirements elaboration.

TABLE V
ELABORATED REQUIREMENTS FOR FIREFOX AND MYLYN

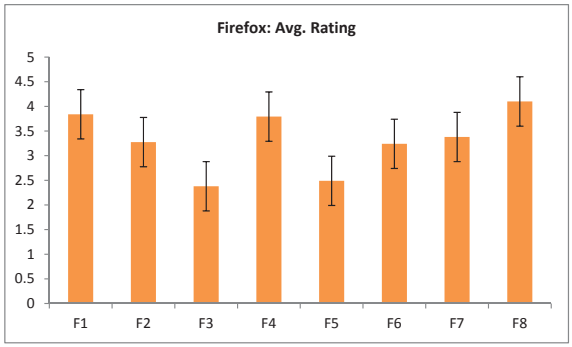
System	Verb	Noun	Elaborated requirement
Firefox	float	view	***F1*:** Firefox should provide **floating** tab-**view**
	button	save	***F2*:** Firefox should provide **button** for **saving** all tabs
	inform	count	***F3*:** Firefox should **inform** user the **count** of open tabs
	select	save	***F4*:** Firefox user can **select** text **saving** directly into files
	arrow	browse	***F5*:** Firefox should provide **arrow** to **browse** tabs
	zoom	drag	***F6*:** Firefox users can **zoom** in/out by a **dragging** slider
	text	number	***F7*:** Firefox user can **text** phone **number** from the web page
	parallel	view	***F8*:** Firefox can **parallel tab**-view
Mylyn	person	set	***M1*:** Mylyn should provide options to **personalize settings**
	window	manage	***M2*:** Mylyn should provide **window** for query **manager**
	credential	issue	***M3*:** Mylyn should be **credentialing issue** tracking system
	plug	comment	***M4*:** Mylyn should **plug comment** to issue tracking system
	shortcut	track	***M5*:** Mylyn should provide **shortcut** to issue **tracker**

information was also collected at this stage through a pre-study survey. The information included software development experience, familiarity with the subject systems, and the primary and secondary programming languages. The recruits reported a median of 3.5 years of software development experience. All the participants have had experience with Firefox (27 users only and 2 contributors) and 9 had knowledge about Mylyn. Irrespective of experience, a tutorial on the latest versions of Firefox and Mylyn was presented. Then the participant was given hard copies of the requirements along with the word pairs (cf. Table V) and was free to use the Internet for further information, if needed.

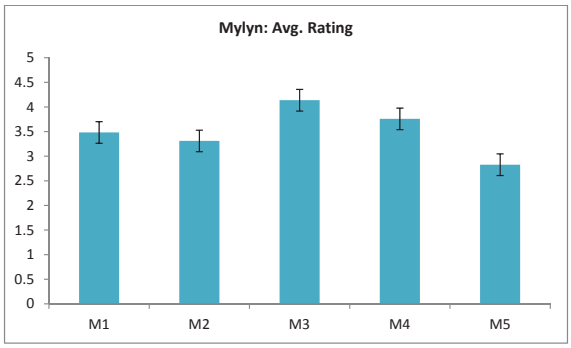
The participant's task was to rate how creative each requirement was by using a 5-point Likert scale: 1=least innovative, 2=not innovative, 3=neutral, 4=innovative, 5=most innovative. It was explained that being innovative in this context meant: 1) Novel and new, as well as, 2) Relevant and useful for the intended software product. The participant was given an option to modify the requirement preserving the given term pair in case she felt some requirement to be less innovative. We asked every participant to work individually on all the requirements. A researcher was present to explain the study, to encourage the participant to think aloud during her session, to take notes, and to conduct an informal exit interview to elicit feedback about her experience. Each participant spent approximately 1.5 hours analyzing all the requirements and was presented with a $10 gift card at the end as a token of our appreciation.

B. Results and Analysis

Figure 6 plots the average ratings reflecting how creative the participants perceive the requirements to be. The average ratings vary from being not innovative (e.g., $F3$ and $M5$) to innovative (e.g., $F8$ and $M3$) based on the 5-point Likert scale (cf. Section V-B). Overall, however, 6 Firefox and 4 Mylyn requirements can be considered innovative based on the average ratings. In order to assess the agreement among the participants on their ratings, we adopt *kappa statistic* (κ), a widely used measure of inter-rater reliability [45]. Kappa statistic returns a value in $[0, 1]$, where κ=0 shows no agreement and κ=1 suggests complete agreement. We find the average κ for Firefox and Mylyn requirements to be 0.79 and 0.67 respectively. According to the magnitude guideline provided by Manning *et al.* [45], these values indicate substantial



(a)



(b)

Fig. 6. Average ratings for the created requirements.

agreement among the participants. These results suggest that *our framework helps generate innovative requirements in an automated manner*. Some individual κ values, however, indicate slight agreement about the innovative values. Section VI provides further insight about this observation.

VI. Discussion

So far, we have discussed how our combinational creativity framework can be applied to create new requirements for Firefox and Mylyn and our assessment on the innovative values of these requirements. In this section, we shed light on some observations and some lessons learned.

A. On Requirements Elaboration from Word Pairs

Compared to the possible idea combinations initially found (1,239,150 for Firefox and 29,864 for Mylyn), our framework has come up with a substantially smaller number of unfamiliar idea-pairs for further elaboration (cf. Table IV). We consider it to be the greatest strength of this framework as requirements elaboration is largely human intensive and a high number of recommended possibilities will make the elaboration phase tedious and overwhelming. Such a laborious activity in a framework like ours will break its main objective of offering automated support. However, the adjustable calibrations during the filtering phase always provide rooms to obtain higher number of unfamiliar idea combinations, if necessary.

One issue though, about our framework, pointed out by Bob (cf Section IV), is sticking to the idea of verb and noun (object) the way they are recommended. We observe Bob spending quite some time to come up with requirements descriptions that preserve the given POS of the words. In several cases, although he started with the verb-noun outline, he rephrased the requirements (e.g., $F2$ and $M5$) to make better senses out of them. When asked about his feedback on the elaboration process during the exit interview, one comment from Bob substantiates our observation.

> I think the verb-object concept was helpful in the beginning to get some initial idea about a requirement. But strictly following that.... I don't think it will always work, nor I think it is necessary. In fact, sometimes it feels really over killing.

We would like to emphasize at this point that preserving the given POS should be considered as an initial guideline and may not be followed as a strict rule. Thereby our framework should provide enough space to phrase the requirements in a less restrictive manner.

B. On Innovative Requirements

Low average ratings: We observe some requirements received relatively low average ratings during the human subject evaluation. For example, $F3$ and $F5$ both received an average rating below 2.5 and based on the corresponding κ statistics (0.83 and 0.65), there was a strong sense of agreements among the participants. In case of $F3$ (Firefox should inform user the count of open tabs), we heard a common argument regarding the usefulness of the requirement to the general users. One participant recommended "Instead, the count of tabs could be provided only if there are many tabs opened at once". For $F5$, most of the participants regarded it to be a redundant one. One participant recommended to make it an optional feature as she commented,

> This requirement would not be very important to a user with a mouse that can easily click on each tab. However, it could be an optional one that is enabled if the user is using a touch screen device.

Requirements with higher ratings for innovation: Some requirements enjoyed higher average ratings with almost perfect agreements [45], e.g., in case of $F8$ (Firefox can parallel tab-view), the average rating was 4.1 (κ=0.81). Some participants, even after rating it 'most innovative', were inspired to brainstorm around the parallel tab-view idea and enriched the requirement with further creative thoughts. The following statements from a participant (rated $F8$ as 5) demonstrates this creativity instigating aspect of our framework.

> It can be made as a button that will split the view of the window and will allow the user to drag tabs into each of them. Also a tab change shortcut can be applied to whichever window is being hovered over, allowing the user to retain the same full tab inventory.

Higher average ratings with limited agreements: In case of a couple of requirements, we observed relatively high creativity ratings but with slight agreements [45] among

the evaluators. $M4$ is such an example (κ=0.19) where two participants identified it to be a particularly useful feature as they perceived it would reduce overhead due to context switches. One participant commented, "As I often contribute to a couple of open source projects, I think this feature will be useful for a user in posting comments and artifacts directly from the IDE". However, many participants identified it as an over killing feature mentioning that, "All the issue tracking systems already provide facilities to post comments..... Don't see any point of implementing it here".

The overall findings attest the creative ability of our framework. In addition, the qualitative results obtained from the evaluation study indicate that the new requirements can also act as starting points and ignite improved creative thinking among analysts. This advances the innovative attributes in the requirements even further, thereby reinforcing the practicality of our framework.

C. Limitations

The work presented in this paper contains the development and demonstration of a conceptual framework, as well as a human subject evaluation. We discuss the limitations from both the framework and the evaluation related aspects.

From the framework perspective: Our framework is limited to its dependency on a large number of existing requirements preferably contributed by a diverse groups of stakeholders. The framework, as it is currently outlined, may not be applicable for a completely new software system in an emerging application domain. Furthermore, applying this framework to a system still at an infant stage may result in fairly limited outcomes. Our framework also largely depends on creating stakeholders' social network that presents a reliable projection of their social interaction. As there exist several social network building techniques (cf. Section II-B), choosing the right means may be tricky. Our framework does not apply any restriction on how the social network should be built and we expect no further limitation along this line. Similar reasoning is also applicable for the network clustering techniques. Clustering the requirements and comments directly (e.g., [49]), instead of the social network, could be an alternative. We believe, however, considering the social network better reflects the collaborative nature of RE [25].

The limitations of topic modeling and POS tagging, such as working with a properly refined and grammatically well written text corpus [16], [17], are also relevant to our framework. However, the requirements and comments the framework tends to analyze are expected to be described in reasonably parsable statements. We also filter out unnecessary and most frequently used words before applying topic modeling. Therefore, we do not expect additional limitations due to these techniques. The practical implementation of our framework revealed a small number of requirements for the subject systems. This is mainly due to time constraints for the analyst and a more conservative filtering criterion (cf. Section IV-B). Further relaxed approach during these activities should help identify a higher number of new requirements.

From the evaluation perspective: An important threat to internal validity is related to the skills of the analyst who formulated the final requirements. It is therefore unknown how different RE skills can impact the evaluation results. We measure how innovative a new requirement is by taking an average of ratings at a 5-point Likert scale and also use the kappa statistics that show substantial agreement among the ratings. However, the participant's level of familiarity with the subject system (specially in case of Mylyn [18]) might have a potential bias in the overall ratings. We mitigate this issue by presenting a tutorial about the subject systems and allowing access to the internet and discussion with the researcher conducting the study for further clarification. We have noticed frequent use of these resources by the participants, thereby improving the reliability of our findings.

VII. Conclusion

In this paper, we have contributed a novel framework that provides automated support for innovating requirements from a combinational creativity perspective [2], [5]. We have also presented a human subject study evaluating how effectively our creativity framework contributes novel and appropriate requirements for an intended software system. The results show that our framework successfully generates creative (i.e., original and relevant) requirements in an automated manner. Furthermore, the new requirements provoke creative thinking of an analyst, thereby improving the innovative aspects.

In future, we plan to reduce our framework's dependency on existing requirements in order to expand its applicability to new software systems and live projects. We are also interested in using methods such as requirements template to facilitate the last phase of our framework, i.e., elaborating and refining requirements, thereby further increasing the automation degree. Finally, we intend to push our framework towards the dimension of transformational creativity in RE [12].

Acknowledgment

We express our sincere gratitude to all the participants for their valuable contributions to the study. This research is partially supported by the U.S. NSF (National Science Foundation) Grant CCF-1238336.

References

[1] J. Lemos, C. Alves, L. Duboc, and G. N. Rodrigues, "A systematic mapping study on creativity in requirements engineering," in *Proceedings of the Annual ACM Symposium on Applied Computing (SAC)*, 2012, pp. 1083–1088.

[2] N. Maiden, S. Jones, K. Karlsen, R. Neill, K. Zachos, and A. Milne, "Requirements engineering as creative problem solving: A research agenda for idea finding," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2010, pp. 57–66.

[3] N. Maiden, C. Ncube, and S. Robertson, "Can requirements be creative? Experiences with an enhanced air space management system," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2007, pp. 632–641.

[4] R. J. Sternberg, *Handbook of creativity*. Cambridge University Press, 1999.

[5] M. A. Boden, *The creative mind: Myths and mechanisms*. Routledge, 2003.

[6] N. Maiden, A. Gizikis, and S. Robertson, "Provoking creativity: Imagine what your requirements could be like," *IEEE Software*, vol. 21, no. 5, pp. 68–75, 2004.

[7] N. Maiden, S. Manning, S. Robertson, and J. Greenwood, "Integrating creativity workshops into structured requirements processes," in *Proceedings of the ACM Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, 2004, pp. 113–122.

[8] N. Maiden and S. Robertson, "Integrating creativity into requirements processes: Experiences with an air traffic management system," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2005, pp. 105–114.

[9] I. K. Karlsen, N. Maiden, and A. Kerne, "Inventing requirements with creativity support tools," in *Requirements Engineering: Foundation for Software Quality*. Springer, 2009, pp. 162–174.

[10] K. Zachos and N. Maiden, "Inventing requirements from software: An empirical investigation with web services," in *Proceedings of the Int'l Requirements Engineering Conference (RE)*, 2008, pp. 145–154.

[11] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher, "Supporting domain analysis through mining and recommending features from online product listings," *IEEE Transactions on Software Engineering*, vol. 39, no. 12, pp. 1736–1752, 2013.

[12] N. Maiden, "Requirements engineering as information search and idea discovery (keynote)," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2013, pp. 1–1.

[13] R. S. Burt, "Structural holes and good ideas," *American Journal of Sociology*, vol. 110, no. 2, pp. 349–399, 2004.

[14] P. Pirolli, "An elementary social information foraging model," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2009, pp. 605–614.

[15] E. Linstead, C. Lopes, and P. Baldi, "An application of latent Dirichlet allocation to analyzing software evolution," in *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, 2008, pp. 813–818.

[16] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[17] E. Brill, "A simple rule-based part of speech tagger," in *Proceedings of the Workshop on Speech and Natural Language*, 1992, pp. 112–116.

[18] M. Kersten and G. Murphy, "Mylar: A degree-of-interest model for IDEs," in *Proceedings of the International Conference on Aspect-Oriented Software Development (AOSD)*, 2005, pp. 159–168.

[19] M. Suwa, J. Gero, and T. Purcell, "Unexpected discoveries and S-invention of design requirements: Important vehicles for a design process," *Design Studies*, vol. 21, no. 6, pp. 539–567, 2000.

[20] M. L. Maher, K. Brady, and D. H. Fisher, "Computational models of surprise in evaluating creative design," in *Proceedings of the International Conference on Computational Creativity (ICCC)*, 2013, pp. 147–151.

[21] G. Ritchie, "Assessing creativity," in *Proceedings of the AISB-01 Symposium on AI and Creativity in Arts and Science*, 2001, pp. 3–11.

[22] R. Lutz, A. Patterson-Hine, S. Nelson, C. R. Frost, D. Tal, and R. Harris, "Using obstacle analysis to identify contingency requirements on an unpiloted aerial vehicle," *Requirements Engineering*, vol. 12, no. 1, pp. 41–54, 2007.

[23] C. Salinesi, R. Mazo, D. Diaz, and O. Djebbi, "Using integer constraint solving in reuse based requirements engineering," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2010, pp. 243–251.

[24] A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso, "Model checking early requirements specifications in tropos," in *Proceedings of the Int'l Symposium on Requirements Engineering (RE)*, 2001, pp. 174–181.

[25] M. Mahaux, L. Nguyen, O. Gotel, L. Mich, A. Mavin, and K. Schmid, "Collaborative creativity in requirements engineering: Analysis and practical advice," in *Proceedings of the International Conference on Research Challenges in Information Science (RCIS)*, 2013, pp. 1–10.

[26] D. Damian, S. Marczak, and I. Kwan, "Collaboration patterns and the impact of distance on awareness in requirements-centred social networks," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2007, pp. 59–68.

[27] A. Begel, Y. Khoo, and T. Zimmermann, "Codebook: discovering and exploiting relationships in software repositories," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2010, pp. 125–134.

[28] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, "Tesseract: Interactive visual exploration of socio-technical relationships in software development," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2009, pp. 23–33.

[29] T. Wolf, A. Schroter, D. Damian, and T. Nguyen, "Predicting build failures using social network analysis on developer communication," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2009, pp. 1–11.

[30] H. Asuncion, A. Asuncion, and R. Taylor, "Software traceability with topic modeling," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2010, pp. 95–104.

[31] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi, "Mining concepts from code with probabilistic topic models," in *Proceedings of the International Conference on Automated Software Engineering (ASE)*, 2007, pp. 461–464.

[32] S. Thomas, B. Adams, A. Hassan, and D. Blostein, "Validating the use of topic models for software evolution," in *Proceedings of the IEEE Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2010, pp. 55–64.

[33] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast collapsed Gibbs sampling for Latent Dirichlet Allocation," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 569–577.

[34] E. Linstead, S. Bajracharya, T. Ngo, P. Rigor, C. Lopes, and P. Baldi, "Sourcerer: Mining and searching Internet-scale software repositories," *Data Mining and Knowledge Discovery*, vol. 18, no. 2, pp. 300–336, 2009.

[35] E. Linstead, P. Rigor, S. K. Bajracharya, C. V. Lopes, and P. Baldi, "Mining Internet-scale software repositories." in *Proceedings of the Neural Information Processing Systems (NIPS)*, 2007.

[36] J. Wu, *Advances in K-means Clustering: A Data Mining Thinking*. Springer, 2012.

[37] S. P. Borgatti, M. G. Everett, and L. C. Freeman, *Ucinet for Windows: Software for social network analysis*. Analytic Technologies, 2002.

[38] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2009, pp. 361–362.

[39] J. Chang, J. L. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei, "Reading tea leaves: How humans interpret topic models," in *Proceedings of the Neural Information Processing Systems (NIPS)*, vol. 22, 2009, pp. 288–296.

[40] S. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, and J. Mylopoulos, "On goal-based variability acquisition and analysis," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2006, pp. 79–88.

[41] N. Niu and S. Easterbrook, "Extracting and modeling product line functional requirements," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2008, pp. 155–164.

[42] C. Fillmore, "The case for case," in *Universals in Linguistic Theory*, E. Bach and R. Harms, Eds. New York: Holt, Rinehart and Winston, 1968, pp. 1–88.

[43] E. Hill, *Integrating natural language and program structure information to improve software search and exploration*. PhD. Thesis, University of Delaware, 2010.

[44] S. Gupta, S. Malik, L. Pollock, and K. Vijay-Shanker, "Part-of-speech tagging of program identifiers for improved text-based software engineering tools," in *Proceedings of the International Conference on Program Comprehension (ICPC)*, 2013, pp. 3–12.

[45] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1.

[46] S. Zaman, B. Adams, and A. E. Hassan, "Security versus performance bugs: A case study on firefox," in *Proceedings of the Working Conference on Mining Software Repositories (MSR)*, 2011, pp. 93–102.

[47] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National academy of Sciences of the United States of America*, pp. 5228–5235, 2004.

[48] N. Niu, J. Savolainen, T. Bhowmik, A. Mahmoud, and S. Reddivari, "A framework for examining topical locality in object-oriented software," in *Proceedings of the Annual Computer Software and Applications Conference (COMPSAC)*, 2012, pp. 219–224.

[49] N. Niu and A. Mahmoud, "Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2012, pp. 81–90.