Requirements Eng (2015) 20:253–280
DOI 10.1007/s00766-015-0226-2

RE 2014

Leveraging topic modeling and part-of-speech tagging to support combinational creativity in requirements engineering

Tanmay Bhowmik[1] · **Nan Niu**[2] · **Juha Savolainen**[3] · **Anas Mahmoud**[4]

Received: 16 October 2014 / Accepted: 6 April 2015 / Published online: 28 April 2015

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 study 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 automatically generates requirements by obtaining unfamiliar idea combinations by means of flipping the part-of-speech of identified topics. We apply our framework on two large-scale open-source software systems (Firefox and Mylyn) and report two studies to assess the viability of combinational creativity in RE. The results show that the creativity merit of requirements generated by our framework judged by human experts is comparable to that of requirements created manually. Meanwhile, the cost of our framework is significantly less than manual work, measured by time spent generating requirements. Our work illuminates a possible improvement toward interactive generation of creative requirements using mechanism's outputs.

Keywords Requirements identification · Creativity · Stakeholders' social network · Social clusters · Topic modeling · Part-of-speech tagging

1 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

✉ Tanmay Bhowmik
bhowmikt@gmail.com; tb394@msstate.edu

Nan Niu
nan.niu@uc.edu

Juha Savolainen
JuhaErik.Savolainen@danfoss.com

Anas Mahmoud
mahmoud@csc.lsu.edu

[1] Mississippi State University, Mississippi State, MS, USA

[2] University of Cincinnati, Cincinnati, OH, USA

[3] Danfoss Power Electronics A/S, Sydjylland, Denmark

[4] Louisiana State University, Baton Rouge, LA, USA

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 focuses more on creativity.

In order to aid creativity in RE, recent research has investigated several approaches. Maiden et al. [3, 6–8] 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 also 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 our previous work [13], we proposed a novel framework to mine ideas familiar to the stakeholders and create 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 [14, 15]. Accordingly, in order to extract familiar ideas, we mined the requirements commonly discussed by distinct stakeholder groups. To that end, we first grouped the stakeholders by clustering the network created based on stakeholders' social interaction. Then, we obtained ideas in terms of dominant topics [16] by applying Latent Dirichlet Allocation (LDA), the most commonly used technique for topic modeling in natural language processing [17]. We further achieved unfamiliar combinations of the dominant ideas by exploiting part-of-speech (POS) tagging [18]. The last phase of our framework involved a human analyst who elaborated requirements from the unfamiliar idea combinations [13]. We applied our framework on Firefox[1] and Mylyn [19], two large open-source software (OSS) systems, and further conducted a human subject evaluation of our framework.

In this extension of our previous paper, we present a further refined framework that eliminates the involvement of the human analyst, thereby providing a fully automated support for combinational creativity in RE. To assess viability, we apply our refined framework on Firefox (see Footnote 1) and Mylyn [19], and create new requirements following an end-to-end automation. A human subject evaluation of the created requirements shows promising practical implications of our framework. In order to assess the cost and effectiveness of our framework with those of manual methods, we conducted a second study that involves humans performing combinational creativity in RE. Building upon the findings from the initial evaluation of our framework, we provide a set of dominant topics from Firefox (see Footnote 1) to three groups of developers. We examine the processes followed by the developers while creating requirements via combination of the given topics, and also recruit two professionals to evaluate the creativity merit of the generated requirements. The findings of these studies suggest that our framework generates creative requirements in a highly efficient manner, and further indicate the prospect of the mechanism's outputs in iterative creative requirements generation.

The contributions of this extended work lie in an advancement of the current solutions that facilitate creativity practice in RE. Our refined framework provides an end-to-end automated support for combinational creativity and complements existing approaches by generating original and relevant requirements. Another important extension to our previous paper is the human subject study involving developers manually performing combinational creativity in RE, and comparing the performance of our framework against manual methods. This opens new avenues for automated supports to aid combinational creativity in RE. The rest of the paper is organized as follows. Section 2 covers background information and related work. Section 3 introduces our framework. Section 4 describes the creation of new requirements for our subject systems. Section 5 details the human subject studies that evaluate our framework. Section 6 presents further discussion and the limitations of the work followed by Sect. 7 concluding the paper with an outline of our future work.

2 Background and related work

2.1 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) [20]. Meanwhile, an idea is appropriate if it is useful to accomplish a task and can be adapted following

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

the task constraints [4]. According to Maher et al. [21], from a design perspective, an idea can be creative if it instigates surprise in terms of deviation in patterns of outcomes. Maiden et al. [2], however, suggest creativity in RE to be mostly situated creativity, i.e., creating requirements and other outcomes new to project stakeholders but that 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 [22] 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 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 third way 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 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 creativity. Now, let us further consider that the initial constraint on hardware limitation is relaxed and we enlarge the search space toward 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, illustrating instances of transformational creativity.

2.2 On the way to creative requirements

In recent years, creativity has been emphasized in the RE literature. Most notably, researchers have conducted creativity workshops involving collaborative and brainstorming sessions among stakeholders. Several other frameworks and tools have also been presented in order to incorporate creativity techniques and heuristics in a direct or indirect manner. In what follows, we provide an overview of the research on creativity in RE.

Creativity workshops: Maiden and Gizikis [6] conducted creativity workshops encouraging brainstorming and creative thinking among the stakeholders during the requirements process. Maiden et al. [7, 8] further proposed RESCUE, a scenario-driven RE process involving workshops that integrated creativity techniques with different types of use case and system context modeling. A case study was also conducted in which RESCUE creativity workshops were used to discover stakeholder and system requirements for a software system intended for managing flight departures from major European airports [7]. Creativity workshops were organized in order to discover features for a future air space management software system for UK and European airspace [3]. These workshops mostly followed exploratory creativity techniques during the analysis of requirements and emergent system properties [3]. Creativity techniques incorporating such workshops require an end-to-end human involvement and depend heavily on the 'aha moments' of the stakeholders during the brainstorming and creative thinking sessions. Furthermore, without any automated support, such techniques are constrained in their scalability.

Frameworks and tools incorporating creativity techniques: In order to support creative thinking for requirements, Zachos and Maiden [10] developed an algorithm that retrieves web services in domains analogical to a current requirements problem. Using this algorithm, they performed an analogical mapping between hotel reservation and parking space booking and explored online hotel reservation systems to create requirements for the Fiat real-time parking space booking system. A tool-supported approach called Semantic Service Search & Composition (S^3C) was developed that explored existing solutions for enterprise resource planning (ERP) systems and supported consultants in creating requirements and relevant services for ERP projects [23]. These algorithms and tools largely depend on finding the right systems or existing solutions and the requirements engineer's ability to draw proper connections between the requirements while performing the analogical mapping.

Lutz et al. [24] presented an approach that performed KAOS Obstacle Analysis to explore requirements in a

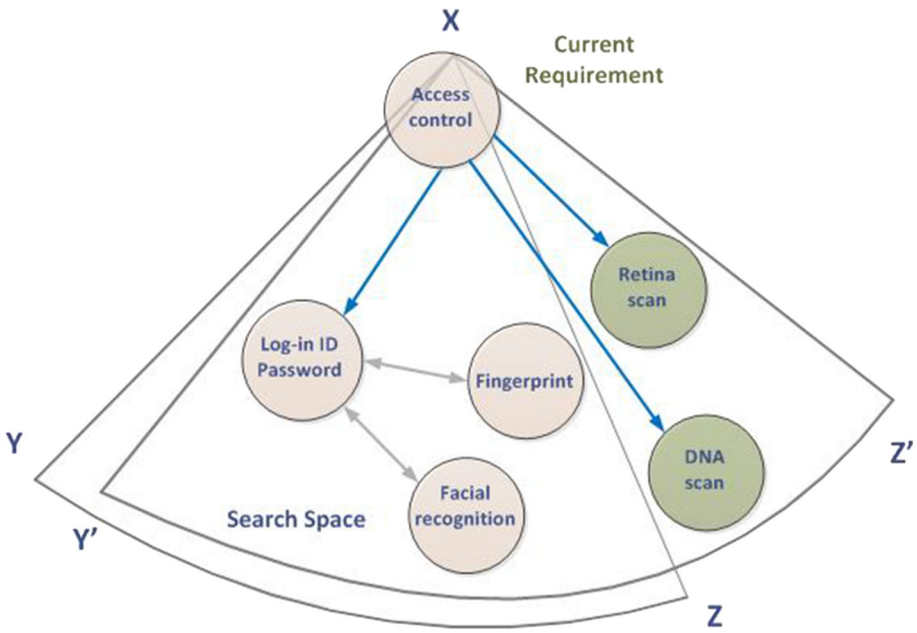

Fig. 1 Categories of creativity based on techniques and heuristics (adapted from [12])

space defined by obstacles for a safety-critical, autonomous system. Salinesi et al. [25] proposed a prototype tool that performed requirement-based product configurations within constraints. This tool discovered various permitted features for a new product in a product line. The *i**/TROPOS approach proposed by Fuxman et al. [26] 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.

Sakhnini et al. [27, 28] applied the elementary pragmatic model (EPM) [29] to support creative requirements elicitation. Their approach focuses on two stakeholders' viewpoints at a time and systematically enumerates the possible combinations of the pair of viewpoints. Two variants were developed: EPMcreate considers all 16 possible combinations and POEPMcreate does the optimization by restricting the enumeration to only four steps which are sufficient to cover the entire space of potential ideas defined by the viewpoint pair [28]. Two controlled experiments using student subjects revealed that POEPMcreate is more effective than EMPcreate, which is in turn, more effective than traditional brainstorming. In both experiments, effectiveness was measured by quantity (raw number of requirements generated) and quality (new and useful). The EPM-based methods, however, rely on requirements engineer's intuition and experience to manually identify the sensible input (namely, two stakeholders) and have scalability concerns by combining only two stakeholders' viewpoints at a time.

Lately, the collaborative nature of creativity in RE has been highlighted by Mahaux et al. [30]. 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 our creativity framework, 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. Moreover, our automated creativity framework reduces the constraints due to human involvement and further improves scalability issues in creativity with the existing approaches.

2.3 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. [31] presented the concept of requirement-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 [32]. In order to create a visual representation for stakeholders' socio-technical relationship, Sarma et al. [33] considered both e-mails among developers and comments in Bugzilla issue tracking system. Posting and reading comments by stakeholders were also considered by Wolf et al. [34] as a means to represent communication flow. Building on the

prior research, we consider in our work posting comments and artifacts on issue tracking systems as social interaction among stakeholders, though our processing results (e.g., topics familiar to a stakeholder group) are clearly constrained by our data sources.

2.4 Topic modeling with Latent Dirichlet Allocation (LDA)

LDA was first introduced by Blei et al. [17] 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 [35–37]. Because the requirements of a software system as well as stakeholders' comments typically contain textual descriptions, LDA becomes particularly useful for our framework. Such textual content can be analyzed to produce latent topic structures for the requirements where every requirement 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 document 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 $d = \langle w_1, w_2, \ldots, w_n \rangle$. 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 [38], 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 $\langle w_1, w_2, \ldots, w_n \rangle$, 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 $\langle t_1, t_2, \ldots, t_n \rangle$. 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) [16, 37, 39], requirements traceability [35], and software evolution [40]. Linstead et al. [16] applied LDA on the source code of different versions in order to analyze software evolution. Linstead et al. [39] 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 [37]. Asuncion et al. [35] 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.

3 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 in a fully automated manner. Being completely automated, rather than relying on human analyst to formulate the new requirements, is a fundamental enhancement of our prior work [13].

Our framework employs five phases to transform the input to the output, which are new and useful requirements. The transformations follow combinational creativity's definition in principle [5, 12], and therefore, carry out two main functionalities: identifying familiar ideas and making unfamiliar connections of those ideas. To that end, ①: The framework takes the software system's existing requirements, together with stakeholders' communication about the requirements recorded in some repository (e.g., an issue tracking system) as input. ②: In order to identify familiar ideas, our basic tenet is that stakeholders of the same social group share common interests [14, 15]. Consequently, the

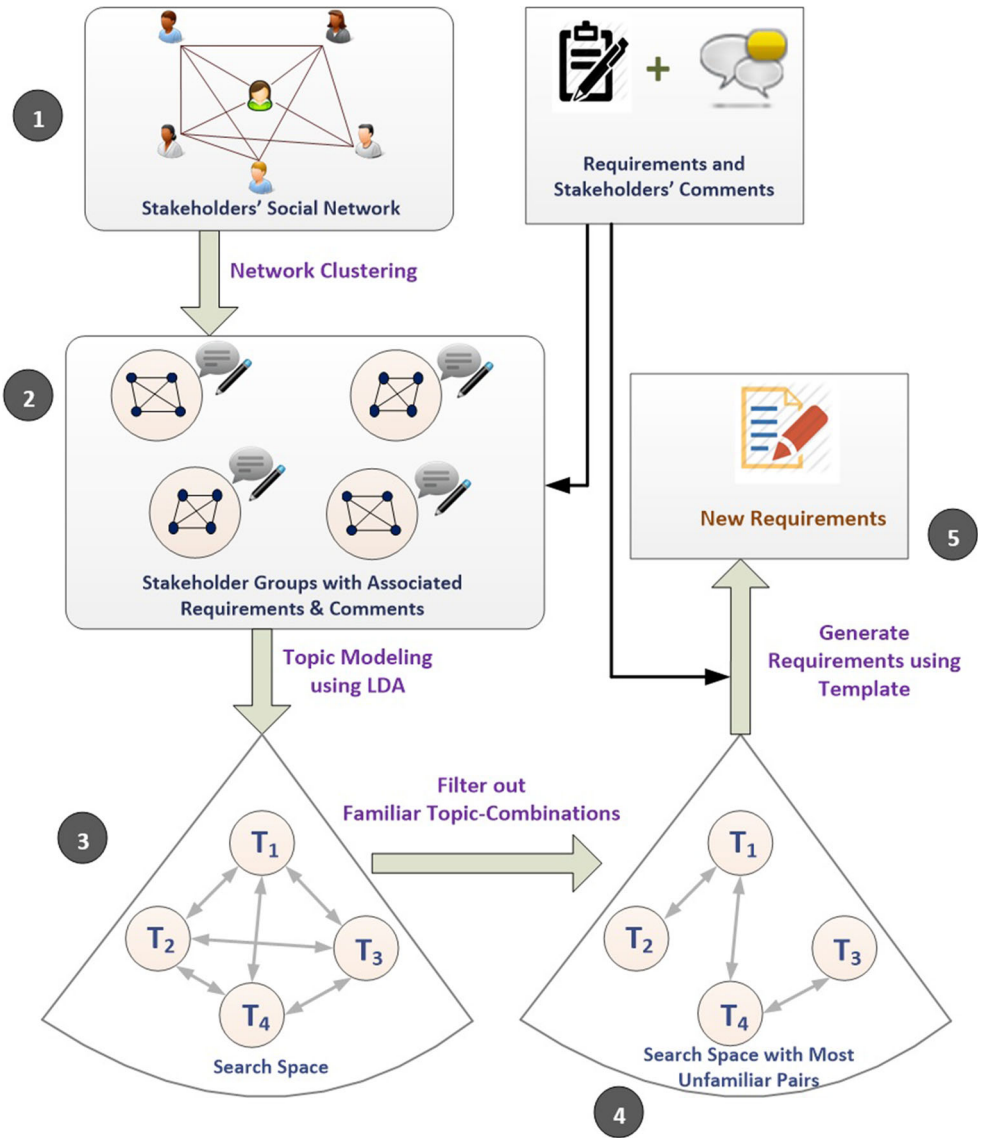

Fig. 2 A framework for combinational creativity

framework clusters stakeholders into separate groups. ③: The thematic topics familiar to each stakeholder group (i.e., familiar ideas) are identified using LDA. ④: In order to make unfamiliar connections of the ideas, the familiar topics' POS (i.e., whether a topic word is being used as a noun or a verb) are tagged. An unfamiliar combination of the topic words are made by flipping their POS (i.e., treating a noun-tagged word as a verb and a verb-tagged one as a noun). ⑤: New requirements are generated from the unfamiliar combination of the topic words using syntactic templates. Next, we describe our framework's five phases in more detail.

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 (refer Sect. 2.3), 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.

Let us consider a hypothetical web browser *B* with 12 stakeholders (A–M). The issues of *B* are recorded in an issue tracking system, and the stakeholders communicate among themselves by posting comments and artifacts over the issue tracker. Therefore, in this scenario, posting comments and artifacts over the issue tracking system

could be considered as the means of stakeholders' social communication. Based on such operationalization, let us assume that Fig. 3 presents the stakeholders' social network for B with 12 nodes and 17 weighted edges. Here, each node represents an individual stakeholder, an edge represents the communication among two stakeholders, and the weight of the edge indicates the number of times those two stakeholders communicate among themselves. Section 4.2 further details the operationalization we use in our study.

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 [15]. Our framework is flexible from the perspective of clustering in that any suitable network clustering algorithm [41] 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 [42, 43]. In the case of the stakeholders' social network for B (refer Fig. 3), a tool such as Ucinet [42] could be used to obtain three social groups as shown in Fig. 4.

3. *Extracting familiar ideas*: As people belonging to the same social group are generally interested in similar ideas [14, 15], 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 [16]) 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 our 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; five to three words were found to be sufficient to convey the topic's subject [44].
- 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.

Figure 5 presents an illustration of determining the cutoff for dominant topics. X-axis represents the topic ID and Y-axis shows the number of times a topic becomes dominant. Taken the cutoff point shown in the figure, topics 0, 1, 5, 30, 41, and 46 are considered to be dominant and familiar ideas. The cutoff line can be tuned based on the trend and the preference of the engineer in order to obtain a reasonable number of topics. Note that Fig. 5 serves as an aid to demonstrate the extraction of familiar ideas in our framework. There should be one such figure for (showing a trend similar to that in Fig. 5) each social

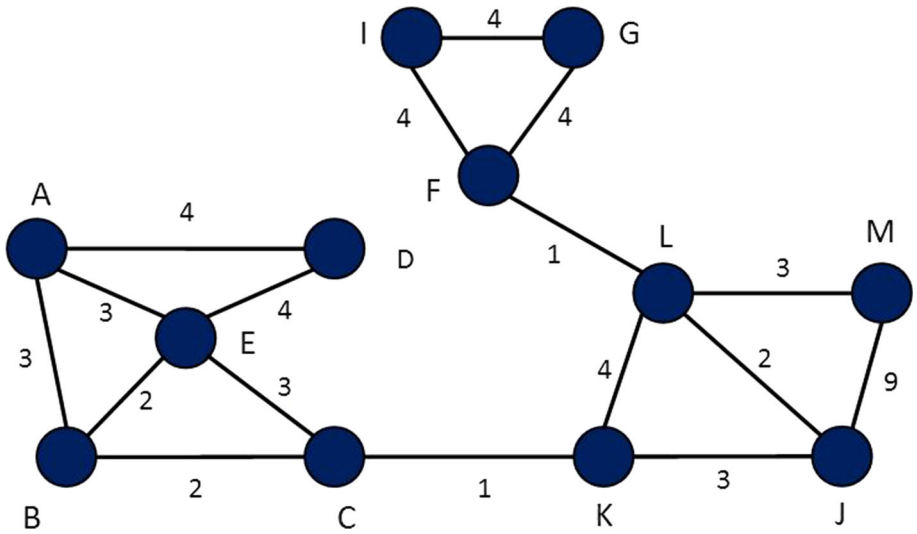

Fig. 3 Stakeholders' social network for the hypothetical web browser B

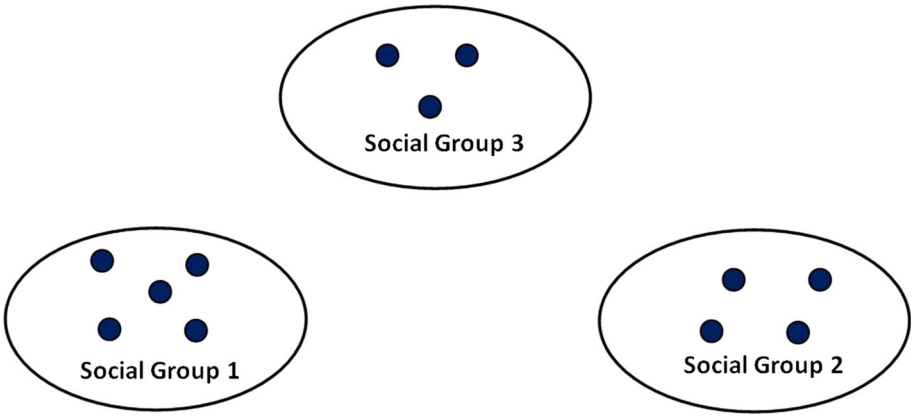

Fig. 4 Stakeholders' social groups for a hypothetical web browser *B*

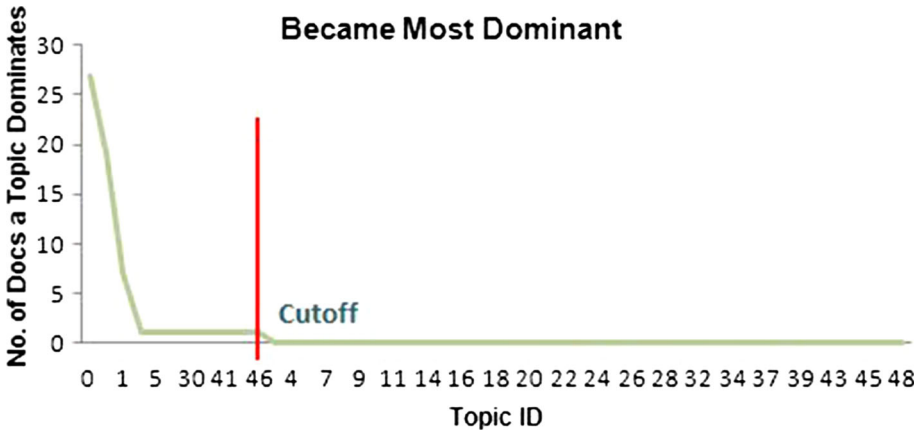

Fig. 5 Dominant topic cutoff

group obtained in the previous phase. For example, while working with our hypothetical web browser *B*, we will obtain three different figures for the three social groups. Following this procedure, the thematic topics familiar to each social group are obtained that serve as an input for the next phase of our framework.

In the case of our example with *B*, let ⟨*zoom, result, browse, context, automatic* ⟩, ⟨*vote, file, block, include, locate* ⟩, and ⟨*sum, mark, start, script, extension* ⟩ be the familiar ideas (i.e., topics) obtained for the social groups 1, 2, and 3, respectively. Note that we choose only one idea for each group for simplicity. In theory, however, there could be more than one familiar idea obtained for each group.

4. Obtaining unfamiliar combinations of familiar ideas: Extracted dominant topics provide us a search space of familiar ideas (refer Fig. 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. Our ultimate goal is to combine these words in an unfamiliar manner, i.e., obtaining word combinations uncommon to the stakeholders. If there are 10 groups with five dominant topics per group and five 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 [45, 46] to tease out the action-oriented theme of a requirement. Such theme, according to Fillmore's case theory [47], 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 two steps: (1) flipping the part-of-speech and (2) finding system specific unfamiliar pairs. These two steps (illustrated in Fig. 6) are designed in such a way that they create unfamiliar combinations of familiar topics with the help of Fillmore's case theory [47], and at the same time, keep the total number of combinations at a manageable level.

- *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 [48] and

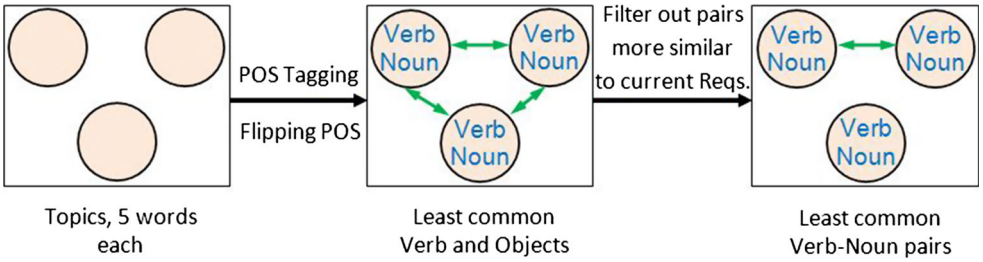

Fig. 6 Finding the least common verb-noun pairs: the leftmost box in the picture contains three topics, represented by *three circles*, with fivewords each

POSSE [49]. We take the most common verb from a topic and the most common noun (object) from another, where two 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 [50] with the current requirements. Then, we filter out the combinations with higher similarity values following a relative filtering approach [50], thereby reducing the search space to most unfamiliar verb-noun pairs (refer Fig. 2).

Let us consider the three topics for browser *B* mentioned in the previous phase. According to the first step in the current phase, assume that we run a POS tagger on the original corpus of existing requirements and comments, and identify the common part-of-speech for each topic word. Let the column 'Common POS' in Table 1 present the common part-of-speech of the topic words. Therefore, we get zoom, vote, and sum as the most common nouns; and browse, block, and mark as the most common verbs for the topics 1, 2, and 3, respectively. After flipping the POS (i.e., considering the nouns as verbs and verbs as nouns), the least common verbs are zoom, vote, and sum, whereas browse, block, and mark become the least common nouns. Thus, the unfamiliar verb-noun pairs are zoom-block, zoom-mark, vote-browse, vote-mark, sum-browse, and sum-block. Note that the verb and noun in a pair come from two topics belonging to two separate groups of stakeholders. At the second step of this phase, we calculate the average textual similarity for each verb-noun pair (i.e., zoom-block, zoom-mark, etc.) with the existing requirements. After we filter out the combinations with higher similarity values (following a relative filtering scheme [50]), let us assume that we obtain zoom-mark as the most unfamiliar verb-noun pair in our example. Section 4.2 provides further details of the relative filtering scheme.

5. Generating requirements from verb-noun pairs: In our previous work [13], our framework included a human analyst, preferably a stakeholder proficient in the software's functional attributes, in this phase. The analyst was provided with all the word pairs obtained from the previous

Table 1 Unfamiliar combinations of familiar ideas for the hypothetical web browser *B*

Topic no.	Topic words	Common POS	Most common		Least common (flipped POS)	
			Noun	Verb	Verb	Noun
Topic 1	zoom	noun	zoom	browse	**zoom**	browse
	result	noun				
	browse	verb				
	context	noun				
	automatic	adjective				
Topic 2	vote	noun	vote	block	vote	block
	file	noun				
	block	verb				
	include	verb				
	locate	verb				
Topic 3	mark	verb	sum	mark	sum	**mark**
	sum	noun				
	start	verb				
	script	noun				
	extension	noun				

Bold fonts indicate topic words

phase as well as some semantic and contextual information about the words. The role of the analyst was to use the word pairs and contextual information (optional) to elaborate requirements in her own words preferably preserving the *ideas* provided by the verb-noun pairs. Our approach was limited in that this phase was likely to be dependent on the analyst's skill and her preconceived knowledge about the system. Mentally processing a verb and a noun that actually looked like a noun and a verb, respectively, was often inconvenient, and working with a good number of apparently discrete words turned out to be exhausting for a human.

Building upon the previous work [13], in our current study, we further eliminate human involvement from this requirements generation phase in order to incorporate complete automation in our framework. To that end, for the noun in each verb-noun pair, we mine the existing requirements and comments and obtain an additional object that most frequently appears along with our noun. Thereby, we extend each verb-noun pair to a verb-clause pair where the clause contains the noun and the object we just mined. Ultimately, the generated requirements will be read by a requirements engineer. Therefore, in the final step, these verb-clause pairs are put into a syntactic template, e.g., subject + verb + clause, in order to generate new requirements.

In the case of our example with browser *B*, let 'area' be the object that appears most frequently along with the word 'mark.' Therefore, 'mark area' is the clause for the verb-clause pair. In order to make the clause further parseable, we add 'ed' after mark, and obtain the requirement 'Browser *B zoom mark*ed area' using the template subject + verb + clause. Note that the verb could also be expanded to a clause using, e.g., adverb and/or frequently used phrases. Our framework remains flexible in that the original verb-noun pair is open to expansion as we have just discussed, and more sophisticated linguistic rules can be applied and a more complex syntactic template can be used to generate the final requirements. Further technical details of the framework are presented in the next section.

The final outcome of our framework is a set of automatically generated requirements. For example, considering the working example discussed in this section, 'Browser *B Zoom mark*ed area' is a concrete outcome. Table 6 presented in the next section provides a complete set of outcomes when we use Firefox and Mylyn as the subject systems for our framework.

It should be noted that our framework utilizes information recorded in the issue tracking system (including existing requirements and comments) to create new requirements. In their seminal work on just-in-time RE, Ernst and Murphy [51] emphasize on the extensive use of issue tracking systems in requirements elicitation and elaboration for OSS systems. Such a use of issue trackers has also been corroborated by other studies involving OSS systems [52–55]. However, our approach is general enough, and other sources of information, such as e-mail and IRC, could also be used in our framework.

4 Creating requirements using our framework

This section explains our procedure of examining how the proposed framework supports combinational creativity in RE. Note that this section no longer dispossesses any working example, rather it details an actual study that we carry out by applying our framework to create requirements for two large-scale OSS systems. In particular, we detail the activities we perform to tease out original, unexpected, useful, and adaptive requirements following the specific phases discussed in Sect. 3.

4.1 Methodology

In order to test our framework, we select two OSS systems: Firefox and Mylyn [19]. 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 [19, 56]. 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 [19]. 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 implemented requirements (closed feature requests) 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

[2] http://www.mozilla.org/en-US/firefox/releases/.

[3] http://www.eclipse.org/mylyn/about/.

information, directly available from the requirements page, is collected by running a Web scraping tool written in Java. Figure 7 illustrates a typical page for a requirement in the Bugzilla issue tracking system. Table 2 presents the collected data that we analyze for the subject systems. Note that we observe many requirements marked as duplicates (especially in the case of Firefox), and exclude them from our study.

4.2 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. [34]. 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. [34], 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 [42], which provides social network clustering and related features. Ucinet [42] 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 [42]. 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 3 presents the clustering results.

Familiar ideas from stakeholders' groups: Phase 3 of our framework applies LDA [17] 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 [57]. From the topic-word matrix and document-topic matrix produced by JGibbLDA, we extract the dominant topics following the heuristics presented in Sect. 3. Along with these matrices, JGibbLDA also produces a topic-word file from which we pick the top five 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 the case of Firefox and 'task' in the case of Mylyn. In such cases, the word is assigned to the topic where it shows the highest probability of occurrence. For instance, the word 'browse' appears in seven different familiar ideas in the case of Firefox. A useful feature of JGibbLDA is that it provides the probability of occurrence for each word in a topic. We compare seven such probabilities for 'browse' and assign the word to the topic where its probability of occurrence is the highest. The results after this phase are summarized in Table 4.

The column 'Possible unique word pairs' in Table 4 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. 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.

Unfamiliar idea combinations for Firefox and Mylyn: This phase first uses Brill's tagger [18] 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-word 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 [58], 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 *average similarity* $\leq 0.15 * \textit{highest similarity}$ and consider them as the final set of verb-noun pairs. Note that the level of this cutoff is subject to calibration depending on the estimated number of requirements an engineer would like to generate using our framework. For example, in [13], we used 20 % of the highest similarity as the cutoff line. However in this paper,

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

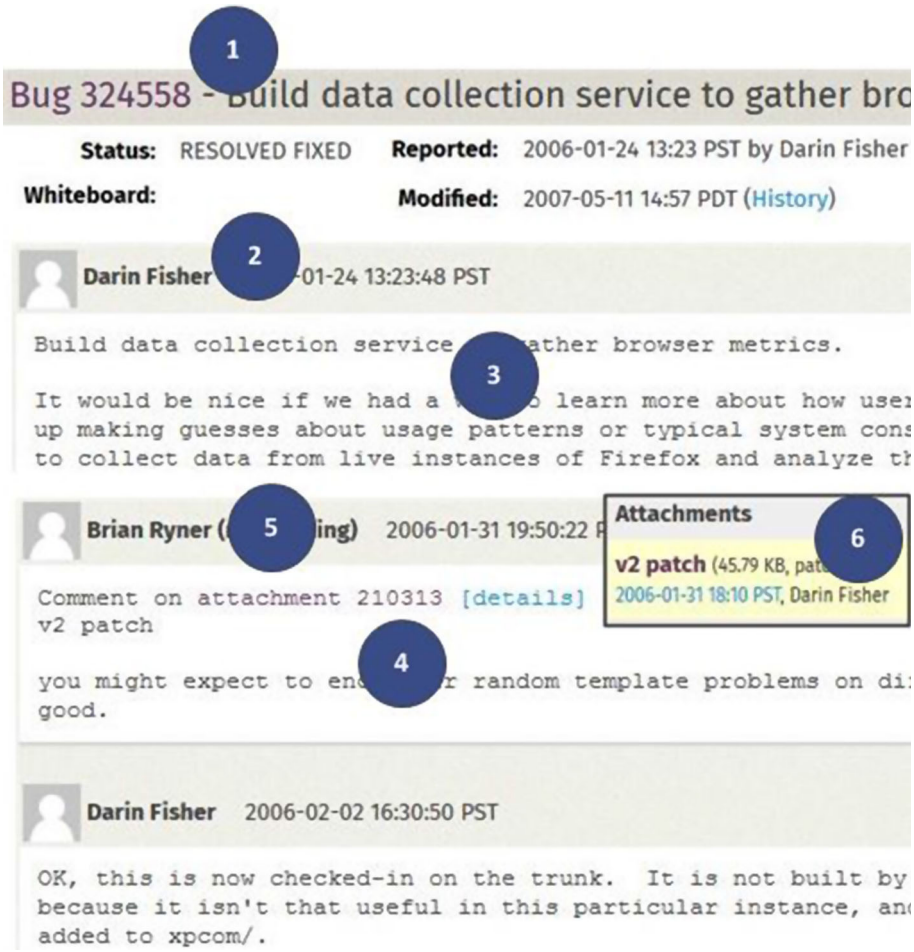

Fig. 7 A Firefox requirement in Bugzilla—some contents are omitted, truncated, and rearranged. *1* Requirement ID. *2* Proposer. *3* Description. *4* Comment. *5* Stakeholder posting comment. *6* Stakeholder posting artifact

Table 2 Data collection of subject systems

System	Application domain	Analyzed history	# of reqs.	Avg. # of comments per req.	# of code files	Written in
Firefox	Web browser	2004–2011	983	18	1968 (C/C++)	C/C++, JavaScript
Mylyn	Eclipse plug-in	2005–2012	445	11	2321	Java

Table 3 Clustering results

System	# of stakeholders in social network	# of clusters (groups)	Avg. group size[a]	Fit value
Firefox	783	36	22 (±8.29)	0.783
Mylyn	136	9	16 (±6.82)	0.817

[a] The average value rounded to the next round number

Table 4 Topics obtained

System	# of clusters	Total # of topics	Avg. # of topics per cluster[a]	Possible unique word pairs
Firefox	36	318	9 (±4.15)	1,239,150
Mylyn	9	48	6 (±3.72)	29,864

[a] The average value rounded to the next round number

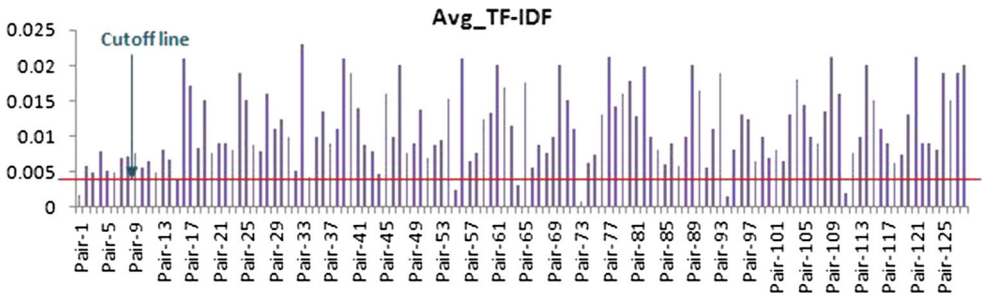

Fig. 8 Finding the least common verb-noun pairs for Mylyn

we follow a more conservative approach and keep the total number of requirements relatively low to better manage the human subject evaluation discussed in Sect. 5. Figure 8 demonstrates the filtering scheme for Mylyn. Table 5 summarizes the results after this phase.

Generating requirements from pairs: The initial objective of this final phase is to extend the noun in each *verb-noun* pair to a clause. In doing so, for both Firefox and Mylyn, the existing requirements and comments are analyzed using Brill's tagger [18], and the object/noun most frequently appeared around each *noun* being considered is picked. The *noun* along with the associated object constitutes a clause, thereby obtaining a *verb-clause* pair. Then, in the final step, a syntactic template is used to generate a requirement formulated from each verb-clause pair.

Figure 9 demonstrates the generation of a new requirement of Firefox from a verb-noun pair. The words 'zoom' and 'mark' were obtained from the topics ⟨*result, browse, context, zoom, automatic* ⟩ and ⟨*sum, start, script, extension, mark* ⟩ respectively. The dotted box contains the additional object 'area' that has been obtained for the noun 'mark.' Thus, 'mark area' becomes the clause, thereby obtaining zoom-mark-area as the verb-clause pair. The next step is to put this verb and clause in a linguistic template to formulate a requirement parseable to humans. Figure 9 details the template we use in this study. Note that our initial template is 'Firefox + shall + verb + clause.' However, in order to obtain a better parseable wording of the requirements, we further formulate the clause in two different ways. The first is adding 'ed' after the noun followed by the object, and the other is 'noun + of + object.' Our intuition is that a requirement worded in either way could be easier to perceive when it is read by a human. Following this heuristic, the final requirement generated by our framework for the pair zoom-mark is 'Firefox shall zoom marked (mark of) area,' in other words, 'Firefox shall provide a feature that allows the user to *zoom* in/out an area she has *mark*ed or selected.' At the end of this phase, our framework finally generates 14 requirements for Firefox and seven for Mylyn, as shown in Table 6.

Compared to our previous work presented in [13], Table 6 is a new contribution in this paper. Here, the last column contains the final requirements generated by our current framework. We observe that some requirements do not sound meaningful and in many cases, the use of 'noun + of + object' in order to elaborate the clause seems to be overkill. Sections 5 and 6 provide further discussion on these issues. Next, we present the evaluation of our framework.

Table 5 Unfamiliar idea combinations

System	Possible idea combinations		
	Initial	After POS tagging	Final
Firefox	1,239,150	2,436	17
Mylyn	29,864	128	7

5 Empirical evaluation of our framework

The overall objective of this section is to assess the viability of combinational creativity in RE. To that end, we describe two controlled studies we performed examining the effectiveness of our framework. In the first study, we evaluated the quality of the requirements generated by the framework. In the second one, we recruited software engineers to manually perform combinational creativity in RE, and further compared the performance of our framework with the manual methods.

5.1 Evaluating requirements generated by our framework

5.1.1 Study setup

The objective of this study was threefold. First, we wanted to obtain an evaluation of the generated requirements. Second, as our current framework provides an end-to-end automation, we wanted to get an assessment of its performance compared to the semi-automated version of the framework presented in our previous work [13]. Third, we wanted to obtain further insights about the effectiveness of our framework through the eyes of a professional. To that end, we recruited a professional software engineer named Bob (pseudonym), working at a local software development company. He possesses about 10 years of

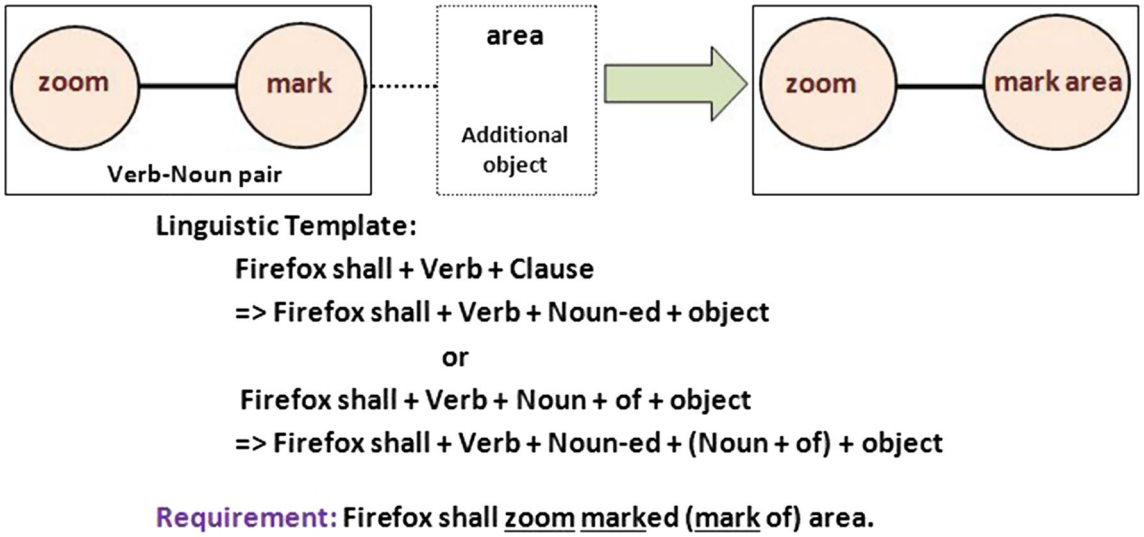

Fig. 9 Example for requirements elaboration

professional experience in software engineering. 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. Note that Bob had also been the human analyst who elaborated requirements from the verb-noun pairs during the requirement elaboration phase in our earlier work [13]. We expected that Bob's experience during our previous study would enable him to make a fair assessment of our current framework with respect to its previous version. Furthermore, our past experience working with Bob made us optimistic about obtaining constructive feedback in light of his professional experience.

We provided Bob with a hard copy of Table 6 and asked him 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 study turned out to be a sort of interview session where a researcher was present to explain the overall objective of the study, to encourage Bob to think aloud while he rated the requirements, to take notes, and to obtain his insight about our framework along various dimensions. It was approximately a 1.5-h session with Bob. In what follows, we detail our findings from this study.

5.1.2 Evaluation results

Figure 10 plots the ratings reflecting how creative Bob perceived the requirements to be. The ratings vary from being least innovative (e.g., $F5, F9, F10, F11, M5$, and $M7$) to most innovative (e.g., $F3$ and $F8$) based on the 5-point Likert scale (refer Sect. 5.1.1). Overall, however, five out of 14 (36 %) Firefox requirements and two out of seven (29 %) Mylyn requirements can be considered innovative based on the ratings provided by Bob. Despite several requirements with very low creativity ratings, the overall picture depicted in Fig. 10 demonstrates the promising outcome of an end-to-end automation for combinational creativity in RE. In what follows, we detail some further insight we obtained about our framework from this human subject evaluation.

5.1.3 Further insight from evaluation

On generated requirements: Bob indicated that a couple of requirements did not make any sense to him even though he went over them over and over (e.g., $F5, F9, F10$, and $F11$), and in a few cases, he perceived the requirements to be already existing, and he provided lower ratings for those requirements. Such an outcome was not very surprising as the requirements were generated by our framework in a fully automated manner without any human intervention. In other words, our current framework had not controlled how meaningful the wording of a created requirement was supposed to be. We posit that the same might be the reason for requirements which turned out to be already existing (or not new). Bob also indicated that he provided a relatively higher rating in the case of a few requirements (e.g., $F4$), even though they did not sound meaningful at the beginning. According to him, revisiting those requirements revealed that they could be reworded in a more clear way, thereby identifying the requirements to be innovative. In his own words:

Table 6 Generated requirements for Firefox and Mylyn

System	Verb	Noun	Additional object from context	Generated requirement
Firefox	inform	match	password	***F1***: Firefox shall**inform matche**d (**match** of) passwords
	float	browse	tab	***F2***: Firefox shall **float browse**d (**browse** of) tab
	note	annoy	user	***F3***: Firefox shall **note annoy**ed (**annoy** of) user
	arrow	browse	tab	***F4***: Firefox shall **arrow browse**d (**browse** of) tabs
	total	drag	browser	***F5***: Firefox shall **total drag**ged (**drag** of) browser
	differ	browse	tab	***F6***: Firefox shall **differ browse**d (**browse** of) tab
	result	document	site	***F7***: Firefox shall **result document**ed (**document** of) site
	zoom	mark	area	***F8***: Firefox shall **zoom mark**ed (**mark** of) area
	button	scroll	window	***F9***: Firefox shall **button scroll**ed (**scroll** of) window
	result	activate	account	***F10***: Firefox shall **result activate**d (**activate** of) account
	sum	match	password	***F11***: Firefox shall **sum match**ed (**match** of) password
	inform	change	setting	***F12***: Firefox shall **inform change**d (**change** of) setting
	float	mark	area	***F13***: Firefox shall **float mark**ed (**mark** of) area
	button	save	selection	***F14***: Firefox shall **button save**d (**save** of) selection
Mylyn	person	set	environment	***M1***: Mylyn shall **person set**ed (**set** of) environment
	widget	post	issue	***M2***: Mylyn shall **widget post**ed (**post** of) issue
	product	cross	platform	***M3***: Mylyn shall **product cross**ed (**cross** of) platform
	window	manage	query	***M4***: Mylyn shall **window manage**d (**manage** of) query
	e-mail	develop	task	***M5***: Mylyn shall **e-mail develop**ed (**develop** of) task
	plug	comment	developer	***M6***: Mylyn shall **plug comment**ed (**comment** of) developer
	patch	update	software	***M7***: Mylyn shall **patch update**d (**update** of) software

Bold fonts indicate topic words

When I first read *F*4, 'Firefox shall arrow browsed (browse of) tab,' I struggled to understand what it meant, even though the terms 'browse tab' caught my attention. Later on I realized that it could be reworded as 'Firefox shall provide an arrow button that will allow the users browse the tabs instead of just scrolling through the tab bar.' I find this to be a pretty cool new requirement.

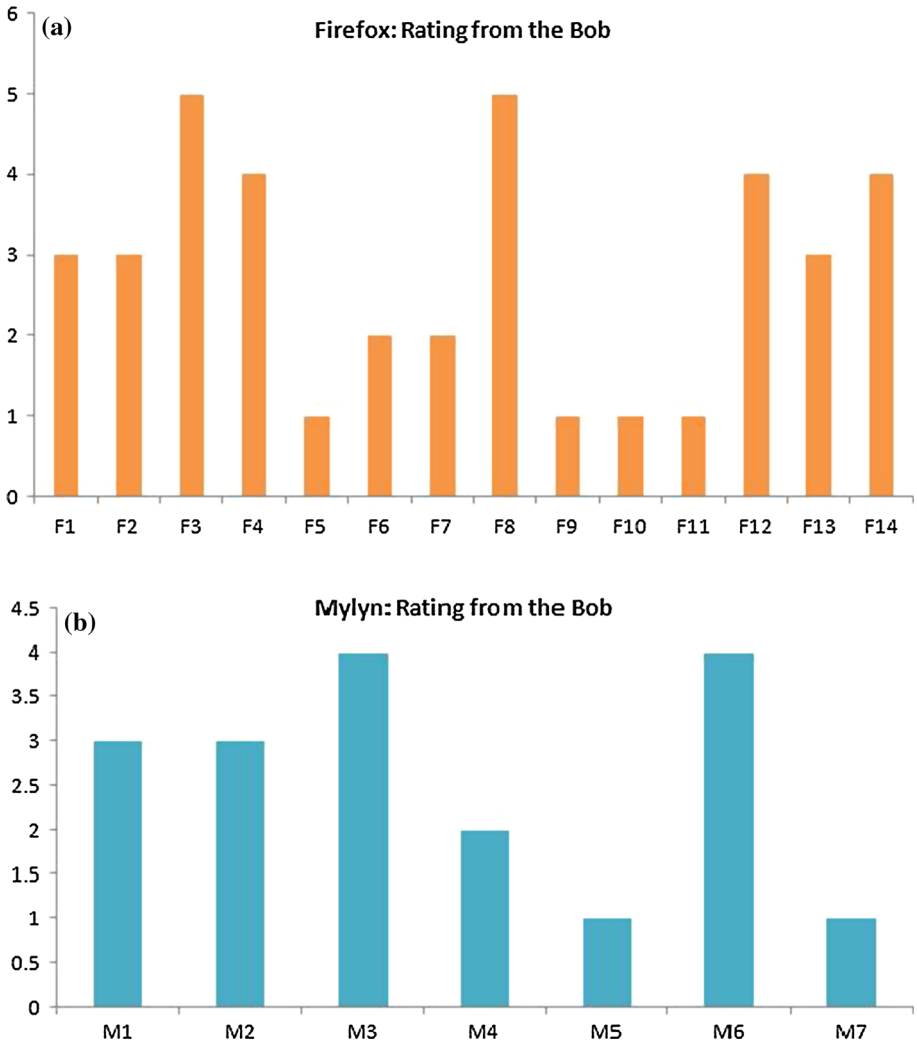

Fig. 10 Ratings by the analyst for the created requirements

In a couple of cases, Bob recalled striking similarity between a requirement (e.g., *M*1) generated by this framework and a requirement he elaborated during our previous work [13] (e.g., Mylyn should provide options to personalize settings). We noticed Bob providing a higher creativity rating (i.e., 4) for *M*1. While pointed out the fact that both the requirements are generated from the same verb-noun pair, person-set, Bob indicated that probably he paid more attention to the contextual information our framework provided while elaborating that requirement. That explains why both requirements expressed similar meanings to him.

We asked Bob about his opinion on using both 'noun + ed + object' and 'noun + of + object' in order to formulate the requirements. He indicated that in most of the cases, the use of 'noun + of + object' seemed to be overkill. In his opinion, however, it was a good approach to have the requirements written in both ways as sometimes one formulation made better sense than the other. According to him:

> I think it is a good thing that you formulated the requirements in both ways. Although more often the 'noun + ed + object' approach looks sufficient, in some cases I think the other way of formulation makes the requirements more promising. For example, when I read the requirement generated from note-annoy pair as 'Firefox shall note annoy of user' instead of 'Firefox shall note annoyed user,' I instantly realize that probably Firefox can have a security related add-on feature that can help the user selectively block pop-ups and ads that she would like to get rid of.

On our creativity framework: Compared to the earlier version [13], the elimination of human involvement from the framework presented in this paper tends to generate relatively higher number of less meaningful requirements. When asked about his opinion about the two versions of our framework, Bob initially indicated this limitation of the current version. However, he provided some further

insights based on his experience with our framework which we find enlightening. First, in our previous work [13], the requirements generated by the framework were largely dependent on the Bob's skill and ability, whereas our current version is not constrained with such dependency. Second, requirements presented in our earlier paper [13] were also influenced by the Bob's preconceived knowledge, and sometimes it had been difficult for Bob to find a work around. Third, being human, elaborating requirements from a large number of apparently unrelated word pairs might often be exhausting and sometimes frustrating, even though the calibration mechanism in our framework could be used to control the total number of word pairs, in Bob's own words:

> I would prefer analyzing some already elaborated statements rather than striving to formulate requirements from scratch.

Bob indicated that in his professional work, he never applied the combinational creativity approach in RE per se. However, in his opinion, any such endeavor by humans should have a baseline to start with. As one of leading goals of any automation is to help humans with exhausting and frustrating repetitive tasks, the testimonials from Bob provide evidence that: (a) our framework works and (b) the extracted common and familiar topics are extremely useful. This indicates the automated support's potential for combinational creativity in RE.

So far in this paper, we have discussed the working mechanism of generating requirements using our creativity framework and presented an empirical evaluation of the quality of the requirements by an expert practicing software engineer. However, this initial evaluation involved only one expert (i.e., Bob), and no other creativity methods were considered and compared with our framework. In addition, lack of sufficient statistical inference limited the generalizability of the findings from the initial study. These limitations indicated the necessity of a further in-depth empirical evaluation of our framework. The rest of this section presents the details of our second study.

5.2 Comparing our automated support against manual methods

In this section, we described our second study that involved software engineers manually performing combinational creativity in RE. We collected data about the quantity and quality of the requirements generated, and further performed quantitative and qualitative analyses comparing the performance of our framework with the manual methods.

5.2.1 Study setup

We recruited nine developers with experience in Java and C#, including both undergraduate and graduate students and staff programmers from one of our institutes. The developers participated voluntarily by responding to an e-mail invitation. We made a confidentiality agreement with the participants to respect their anonymity. The demographic 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 years of software development experience. All the participants had experience with Firefox (eight users only and one contributor) and four had knowledge about Mylyn. Because of the domain unfamiliarity of Mylyn, in order to help the developers keep better focus while performing the activities, we considered Firefox as our subject system for manually generating requirements. Irrespective of experience, a tutorial on the latest versions of Firefox was presented.

As RE naturally involves substantial collaboration among stakeholders, following Paulus and Nijstad [59], we decided to study groups of collaborating developers conducting combinational creativity instead of individuals. To that end, we randomly assigned the developers to three different groups. In this paper, we refer to the groups as A, B, and C respectively. As the initial baseline, we intended to provide the dominant topics our framework used in generating those 14 requirements for Firefox (refer Table 6). Note that the way we operationalized an unfamiliar verb-noun combination, the verb and the noun were originated from two different dominant topics (refer Sect. 3). As a dominant topic could provide one such verb and noun, our framework would require at least two dominant topics to generate two unfamiliar verb-noun pairs. Furthermore, the same verb or noun could be paired with a different noun or verb, respectively, originated from different topics, thereby contributing different verb-noun pairs. As a result, when we traced back the verbs and nouns presented in Table 6, we obtained ten topics associated with those words. Each group of developers were provided with a hard copy of those ten topics detailed in Table 7 where each topic contains five words.

Each group was asked to elaborate creative requirements, as many as they could, by combining words picked from different topics. Note that we did not put any restriction on how many topics the developers could combine. A researcher conducted a tutorial session demonstrating what creativity means in RE and explaining our study to the participants. Each group worked

Table 7 Firefox topics for developer groups

Topic no.	Topic words
Topic 1	reply, float, progress, link, comment
Topic 2	display, skin, tab, difference, match
Topic 3	flash, button, reproduce, activate, add-on
Topic 4	password, account, option, save, result
Topic 5	blank, approach, document, inform, scroll
Topic 6	sum, start, script, extension, mark
Topic 7	statement, arrow, annoy, processor, application
Topic 8	result, browse, context, zoom, automatic
Topic 9	drag, move, note, expect, query
Topic 10	total, special, higher, provide, change

separately. They were allowed to have as many discussion sessions or group meetings as they liked. We asked every group to keep a detailed record of their group meetings and different strategies they followed while formulating requirements from topics. Each group was also requested to mark the requirements they perceived to be more innovative than the rest. The developers were given a time window of one week. At the end of the week, each group submitted a report to a researcher providing the requirements they generated along with a detailed account of their group sessions and different strategies they followed.

We wanted to further compare the cost and effectiveness of our framework with those of the manual approach. Our objective was to investigate if the framework generated creative requirements with a significantly lower cost compared to manual methods. In particular, we aimed to test the following two hypotheses:

- **H1**: Time required to create requirements using our automated framework is significantly less than that of the manual approach.
- **H2**: The quality (i.e., creativity merit) of the automatically generated requirements is significantly higher than that of the manual approach.

Note that in the case of H1 is supported, H2 is stronger than needed. That is, if we find evidence that supports H1, what we really need is the quality (i.e., creativity merit) of the automatically generated requirements to be at least as effective as (or comparable with) the manual approach. However, in formulating H2, we take a more aggressive approach than required.

In order to test H1, for both the automated and manual approaches, we collected data for the following measures.

- *time*: time spent in generating new requirements
- *total*: total number of new requirements generated
- *creative*: total number of creative requirements generated

We monitored the total amount of time our framework required to created requirements in order to get the value of *time* for the automated support. In the case of the manual methods, we obtained the measurements from the report submitted by each group.

H2 deals with the creativity merit of the generated requirements. In an attempt to test H2, we recruited two professional software engineers, named Jim and Sara (pseudonym), working at a local software development company. They possessed, in total, 12 years of professional experience in developing software. As did Bob (refer Sect. 5.1), they both performed requirements analysis-related activities at a regular basis, and were familiar with both Firefox and Mylyn. Jim and Sara were provided with a hard copies of both the automatically generated and manually created requirements. They were asked to rate how creative each requirement was by using a 5-point Likert scale as described in Sect. 5.1. Jim and Sara worked individually and spent, on an average, about an hour in rating the requirements. The study ended with a researcher conducting an informal exit interview with the participants, and presenting a $10 gift card to each of them as a token of our appreciation.

5.2.2 *Manually performing combinational creativity in RE*

We analyzed the reports provided by developer groups as well as the data collected through exit interviews. Two researchers spent, in total, 7 h going through the reports and interview records, and extracting important attributes about the processes developers followed in generating requirements. According to our analysis, each group followed a different process in combining topics and elaborating requirements. In what follows, we present a detailed discussion on each process we identified. In this section, we use the terms participants and developers in a synonymous manner.

5.2.3 *Divide-and-Trade*

Figure 11 provides a general overview of the process group *A* followed while elaborating requirements from topics. The participants in this group first started with reviewing the given topics and obtaining a better understanding of the topic words. We name this review phase as *preprocess*. While preprocessing, group members came to an agreement that each member working with a subset instead of all the topics would help them manage the total number of topics as a group in a more convenient manner. Thereby, in their next phase, the group divided the given topics into three disjoint subsets, and each member picked one subset to start topic combinations. We noticed that, after picking a subset of topics, the group members typically followed two

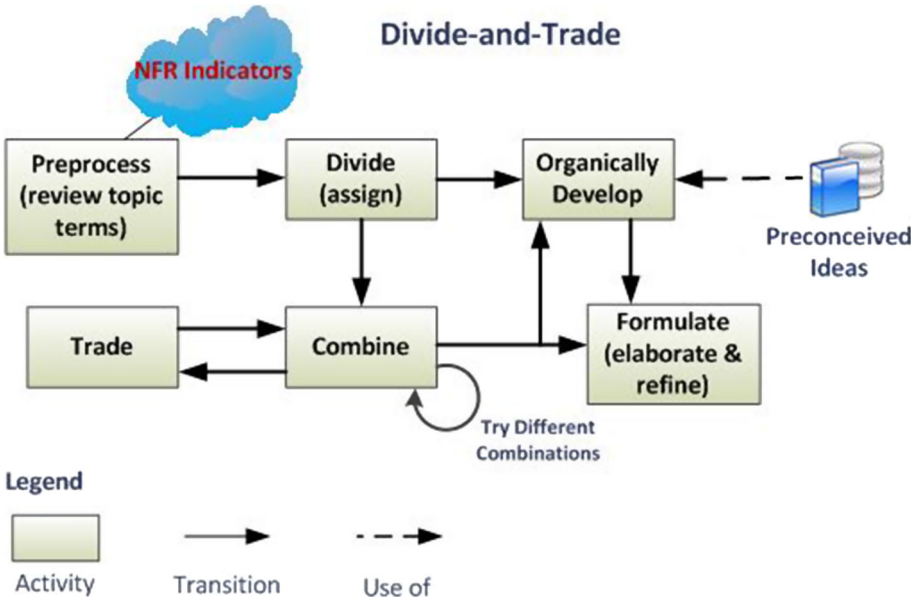

Fig. 11 Divide-and-Trade approach, followed by Group *A*

different methods while creating new requirements which are discussed next.

Combine-trade-formulate: Following this method, the developer attempted different combinations of the topic words and tried to make an intuitive sense. A promising idea generated from such combinations were noted down, shared and discussed with others, and eventually got elaborated and refined into a new requirement, if possible. We call the phase involving elaboration and refinement as *formulate*. This process got repeated until the participant ran out of promising options. According to the report submitted by group *A*, while performing such combinations, each developer chose at least one topic from the subset she was working with, while being free to choose the other topics from the rest of the subsets. After the participants exhausted all their options, they traded their set of topics, and repeated the combining and formulating activities with the new topics.

Organically develop and formulate: Sometimes, inspired by some topic words, a participant started developing a requirement organically with the help of her preconceived idea about a scenario. If a requirement developed in this manner looked promising, it was shared with the other group members. Then, the group reformulated and refined the requirement using specific words from the given topics. Group *A* reported that such organically developed requirements sometimes originated from the group discussions during the combination and formulation of some other requirements.

One interesting finding from the activities of this group is that whenever a participant perceived a topic word to be related to some sort of nonfunctional attribute, she considered that topic word along this line and tried to come up with a relevant nonfunctional requirement. For example, the word 'account' was widely attached with security- and privacy-related concepts by the group members, and the requirements created using this word (e.g., *R*3) support our observation. In Fig. 11, we name such topic words as 'NFR Indicators.' As dividing and trading requirements among the group members play a vital role in the creativity approach followed by group *A*, we identified this process as *Divide-and-Trade*.

5.2.4 *Individual-Interleaved-with-Group*

After analyzing the process pursued by group *B*, we identified two interleaved sets of activities performed by the participants. Figure 12 details different activities in the process. The procedure followed by this group started with each member working individually and handling all ten topics in an isolated manner. In this paper, we call it the *preparation* phase. The participants tried different combinations of the topic words and elaborated a requirement when identified an apparently promising combination. According to the report and the exit interview, the participants heavily used external resources while performing this activity. For example, they browsed Web sites such as YouTube and tried to think about some requirements correlating with the topic words while having a first hand experience with the browser. Each participant recorded every requirement she came up with and prepared for the group meeting.

After all the group members were done with their individual efforts, they set up a meeting to discuss their individual outcomes and to finalize the set of requirements identified as most promising. In doing so, the group further

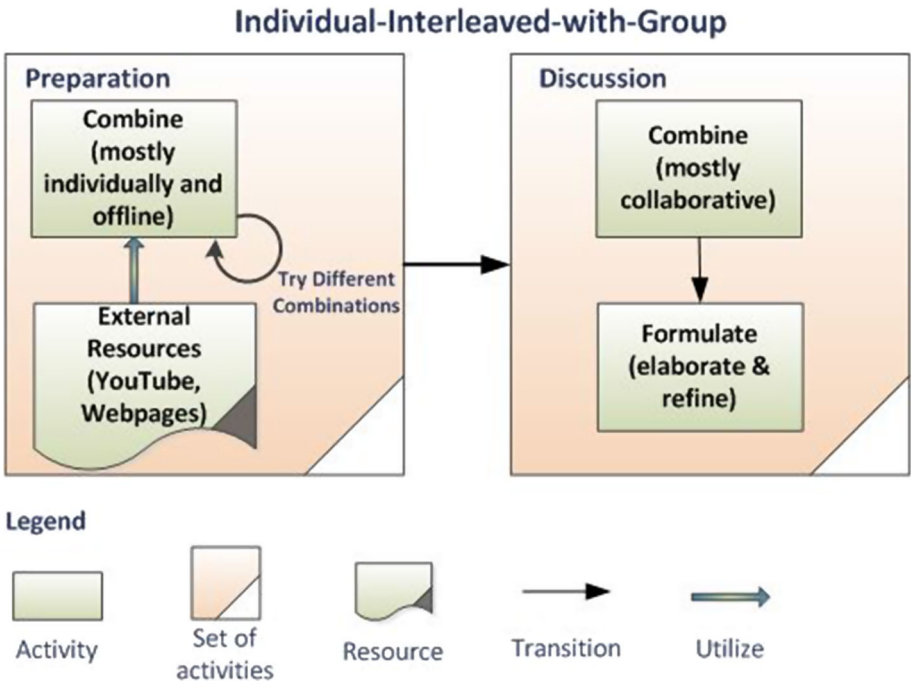

Fig. 12 Individual-Interleaved-with-Group approach, followed by Group *B*

combined the promising topic words, if required, in a collaborative manner, and *formulated* the final set of requirements through elaboration and refinement. As we observe two distinct phases of activities conducted by group *B*, we call their creativity process as *Individual-Interleaved-with-Group*.

5.2.5 Group-Heavy

After analyzing the activities performed by group *C*, we identified that an end-to-end collaboration among the participants played the dominant role in their creativity approach. We call this process *Group-Heavy* (presented in Fig. 13). As one participant mentions during the exit interview:

> Our approach was mostly collaborative. Activities starting from topic discussion to requirements elaboration, we did everything capitalizing on our effort as a group.

The group started with reviewing and discussing the terms in each topic and noted down any insight about the topic the group members perceive to be useful. We indicate these activities as *preprocess*. Then, the group pursued either of the following two different approaches based on their initial insights about the topics.

Combine and formulate: When some topics looked promising to the group, different combinations of the topic words were discussed among the members. The group went through several trials until a potentially meaningful combination is obtained. After a suitable was found, the group expanded the combination in natural language, reformulated, elaborated, and refined the wordings, thereby *formulating* a complete requirement.

Organically develop, retrofit, and formulate: Sometimes, another approach group *C* followed started with organically developing a requirement based on some preconceived idea. In the case of such development, the theme of the requirement was maintained along the theme of some potential topics. Then, terms and phrases were replaced by most suitable topic words keeping the theme of the requirement intact. We name this particular activity as *retrofit*. This version of the requirement was further elaborated and refined, in order to obtain the final *formulation*.

One interesting attribute about the process followed by this group was that they tried to utilize the concept of some *inconvenience* the group members have experienced in their real life. No matter what elaboration approach the group followed, utilizing the concept of inconvenience was recurrent in their process. This finding was also attested by the participants during our exit interview session.

So far, we have discussed the three different processes our human participants followed while obtaining new requirements through combinational creativity. In what follows, we analyze the data collected for different quantitative and quantitative measures, and test the hypotheses we formulated to compare the cost and effectiveness of our automated framework with those of the manual approach.

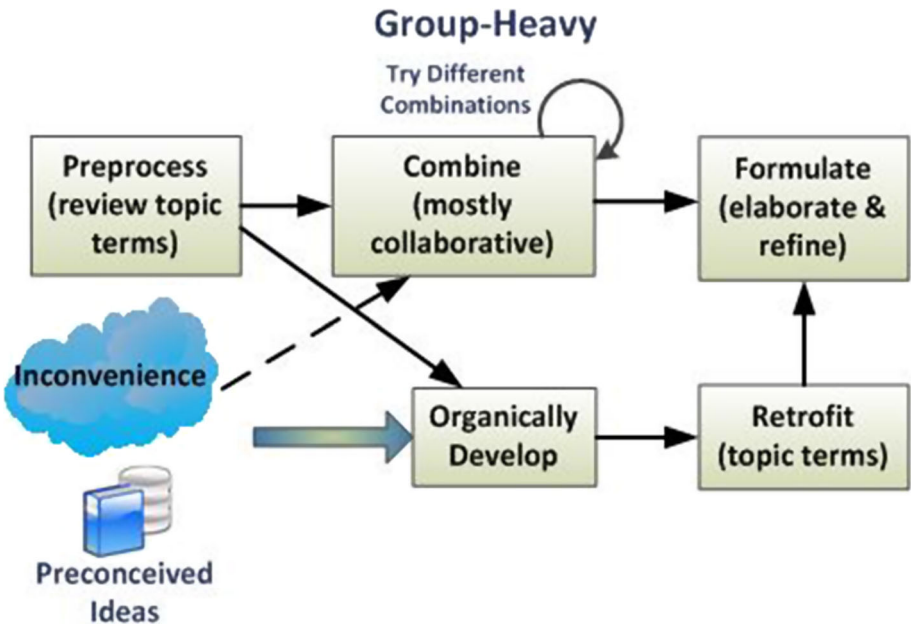

Fig. 13 Group-Heavy approach, followed by Group *C*

Table 8 Comparing our automated approach with manual methods

Approach	System or group	Time spent (time)[a]	Total no. of new requirements (total)	No. of reqs. above threshold according to		Avg. no. of reqs. above threshold (creative)
				Jim's ratings	Sara's ratings	
Automated	System: Firefox	37	14	8	7	7.5 (53.57 %)
Automated	System: Mylyn	22	7	4	4	4 (57.14 %)
Manual	Group: A	150	15	7	9	8 (53.33 %)
Manual	Group: B	300	13	5	4	4.5 (34.62 %)
Manual	Group: C	270	17	9	7	8 (47.06 %)

[a] Measured in minutes

5.2.6 *Results and analysis*

Table 8 summarizes the collected data regarding the automated and manual approaches of generating requirements along with the evaluations by Jim and Sara. The columns 'Time spent (time)' and 'Total no. of new requirements (total)' represent the data for measures *time* and *total,* respectively (refer Sect. 5.2.1). We use these two measures to test our first hypothesis, H1. Recall from Sect. 5.2.1 that each group was also requested to mark the requirements they perceived to be more innovative than the rest. According to the reports submitted by the groups, the number of such innovative requirements are 10, 5, and 11 for the groups A, B, and C, respectively. Tables 9, 10, and 11 detail these innovative requirements indicated by the groups.

In order to test H2, we need to compare the quality of the requirements generated by our automated framework with that of the manual process. To that end, we need data for the measure *creative* as discussed in Sect. 5.2.1. In the case of the manual process, considering the number of innovative requirements indicated by the groups could be an option. However, such a choice imposes two major limitations on the analysis. First, these numbers could be highly biased as the requirements were rated by their creators themselves. Second, we do not have any such explicit classification made by the groups on the automatically generated requirements, thereby making any statistical analysis infeasible. Therefore, we consider the ratings provided by Jim and Sara for both the automated and manually generated requirements in order to obtain data for the measure *creative*. Our operationalization is as follows.

- We consider the individual creativity rating for each requirement.
- As the requirements are rated at a 5-point Likert scale, we pick any requirement with a rating ≥ 3 and classify it as a creative requirement.

Table 9 Innovative requirements (above threshold) according to Group A

Requirements above threshold as indicated by Group A[a]	No. of topics used
A1: The browser shall **provide** similar web pages to the one currently being viewed by **matching** the content on the web pages	2
A2: Loading a new webpage shall **automatically** execute **scripts** that check an **account**-based list of banned words and removes them and their **context** from the text on the new **page**	5
A3: Marking items on a page shall **move** their source to an ignore list that is **saved** to the user **account** and then user will no longer receive any content from that source	3
A4: The execution of a malicious **script** shall cause the **skin** of the browser to **flash** in an alerting manner	3
A5: The browser shall be able to determine the **difference** between the necessary and unnecessary pop-ups based on the **context** of the web page	2
A6: The web browser shall **link** to the user **account** calendar and the **skin** will **flash automatically** when a calendar event is **approaching**	4
A7: When the user highlights **text** and presses a **special button** the browser shall create a **blank document** and **move** the highlighted text to the document	5
A8: User **accounts** shall be able to **save** personal **comments** on a webpage	3
A9: Dragging a **link** to the 'new **tab**' **button** shall **start** a new **tab** at the address of the **link**	6
A10: Clicking and **dragging** an item to a specific icon on the UI shall **save** the selection to a directory on the local machine	2

[a] Topic words are in bold fonts

Table 10 Innovative requirements (above threshold) according to Group B

Requirements above threshold as indicated by Group B[a]	No. of topics used
B1: The web browser shall provide **add-on** to enable text to speech compatibility while reading in **documents** on the web (this will allow the user to continue reading or browsing different **tabs**)	3
B2: The web browser shall **fix** the **context** and provide an appropriate **resul**t depending on parental control **options**	2
B3: The web browser shall **provide** an **option** to have a split **view** to browse two **documents** simultaneously	4
B4: The web browser shall **provide** a **float option** to **view** content on top of any current tab being used	3
B5: The web browser shall **provide** an **option** to change the **view** to color-blind mode	3

[a] Topic words are in bold fonts

The fifth and sixth columns in Table 8 indicate the number of such creative requirements according to the ratings from Jim and Sara, respectively. In order to assess the agreement among raters on their ratings, we adopt *kappa statistic* (κ), a widely used measure of inter-rater reliability [50]. Kappa statistic returns a value in $[0, 1]$, where $\kappa = 0$ shows no agreement and $\kappa = 1$ suggests complete agreement. We find the average κ to be 0.71 and 0.68 for Firefox and Mylyn, respectively, in the case of the automatically generated requirements. For the manually generated requirements, average κ for groups A, B, C and are 0.72, 0.67, and 0.7, respectively. According to the magnitude guideline provided by Manning et al. [50], these values indicate substantial agreement between Jim and Sara. Based on this analysis, we further simplify our data and consider the average number of creative requirements according to Jim's and Sara's rating as the values for *creative* (refer the last column in Table 8). In what follows, we test the two hypotheses we formulated for this study.

H1: Time required to create requirements using our automated framework is significantly less than that of the manual approach. Our automated framework generated 14 new requirements in 37 min for Firefox and seven new requirements in 22 min for Mylyn. Note that major portion of this time was required to create stakeholders' social network which is the most computationally demanding phase of our framework. In the case of the manual approach, groups A, B, and C generated 15, 13, 17 new requirements in 150, 300, and 270 min, respectively. As Firefox and Mylyn are two different systems and the manual requirements generation was performed only for Firefox, we normalize the data in order to make a statistical analysis viable. To that end, we consider the average time spent per requirement. Our data show that the automated framework generated a new requirement for Firefox and Mylyn in 2.64 and 3.14 min, respectively. In the case of the manual approach, the groups A, B, and C spent on an average 10, 15.88, and 23.1 minuets, respectively, to

Table 11 Innovative requirements (above threshold) according to Group C

Requirements above threshold as indicated by Group C[a]	No. of topics used
C1: Drag videos out from web pages, make it **float** on the top of screen	2
C2: Comment box in browser to copy/paste **text/documents** for clipboard synchronization across devices	3
C3: If a user has shopped at a website with security breach, **inform** them that their recent purchase from their **account** may have been compromised and **provide** a **link** to details about the breach	3
C4: Add-on that **expects** your next **move** such as opening a new **tab**, **starting** an **application** by analyzing your **account** activity	6
C5: Social media **button** that opens a scrollable-drop down with all your **accounts** that will allow you to **view texts**, **comments** and **reply** to them	6
C6: Drag text to other pages and **reproduce** a new html page for compared reading	3
C7: Reply to **text** messages on your phone via a synced **accoun**t	3
C8: A custom password settings suite that ensures your **password** meets certain user-defined criteria (i.e., **text**, **numbers**, **special** characters), ensures that both the entered password and the confirm password fields **match**, and prevents **document**/form submission when the password fields are **blank**	5
C9: Use multiple **processors** on the CPU and GPU to provide **higher**-quality **flash** and video content	3
C10: Auto-fill data that are user-editable and location aware for **automatic**, **contextual** form data predictions that are synced to your **account**	2
C11: A **button** for **saving** information to an offline tab with a **link** for later **viewing** which is available on **startup**	5

[a] Topic words are in bold fonts

generate a new requirement. This information is presented in Fig. 14.

Following [27], we perform a two-sample t test for unequal variances [60] in order to test H1. According to the t test, there is a significant difference between the time required to create requirements using our automated framework $(M = 2.89, \text{SD} = 0.35)$ and that of the manual approach $(M = 16.32, \text{SD} = 6.55)$ where $t = -3.54$, $p = 0.04$. In other words, H1 is accepted at $\alpha = 0.05$, i.e., our automated framework requires significantly less time than manual approach.

H2: The quality of the automatically generated requirements is significantly higher than that of the manual approach. Following the rationale discussed while testing H1, we further use normalization in testing H2. To that end, we consider the percentage of the total number of generated requirements being creative instead of the raw number. In the case of our framework, 53.57 and 57.14 % of the generated requirements are creative for Firefox and Mylyn, respectively. Considering manual approach, this measure for the groups A, B, and C is 55.33, 34.62, and 47.06 %, respectively (refer Table 8 and Fig. 15). A two-sample t test for unequal variances [60] indicates that at $\alpha = 0.05$, the quality of the automatically generated requirements $(M = 55.36, \text{SD} = 2.53)$ is not significantly higher than that of the manual approach $(M = 45.00, \text{SD} = 9.52)$ where $t = 1.8$, $p = 0.11$. Therefore, following [27], we conclude that the quality of the requirements generated by our automated framework and that of manual approach *is comparable*.

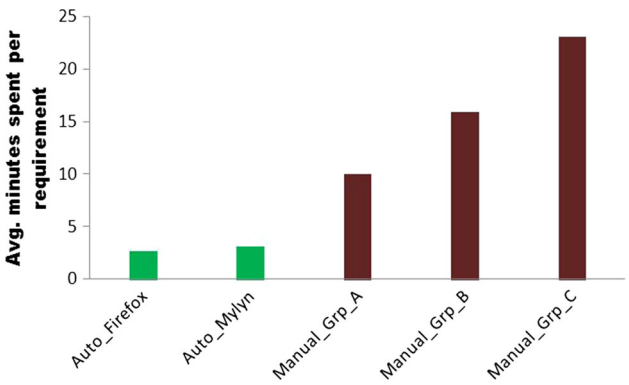

Fig. 14 Average time required to generate a requirement for both the automated and manual approaches

5.3 Threats to validity

Construct validity is the extent to which the empirical study and its various measures test and measure what they claim to test and measure [61]. This kind of validity applies mainly to our second study where we formulated two specific hypotheses. The aim of our second study was to test the cost-effectiveness of our approach. Therefore, the core constructs that we used were time for cost and creativity merit for effectiveness. These measures are in line with Sakhnini et al.'s study [28] where the authors evaluated a creativity enhancement technique for requirements generation. Although the sheer number of generated requirements was considered part of effectiveness in [28],

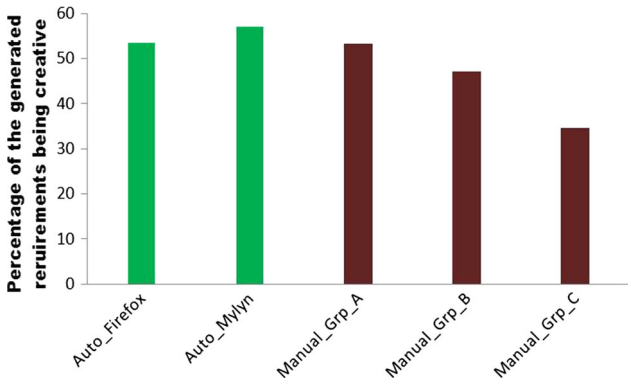

Fig. 15 Percentage of generated requirements being creative

we valued quality judged by experts in a similar way as [28] more than quantity.

Internal validity refers whether one can conclude the causal relationship that is being tested by the empirical study [61]. Again, the validity applies mostly to our second study. As our results show support for H1, we claim that the differences in the adopted methods caused the observed differences in the time spent generating requirements. For the nine subjects participating in our second study, we randomly assigned them into groups with the intention to balance their domain familiarities and experience levels. Although some team spent less time than others (refer Fig. 14), all manual work was more costly than our automated framework. In the future, we plan to take into account factors like individual creativity [28] to better control the confounding variable related to personal or even group differences.

External validity concerns how much the results can be generalized to other cases [61]. Clearly the small number of data points stands in the way of generalization. While we chose two open-source projects to apply our framework, systems from other domains or of proprietary natures may exhibit different characteristics. Our reliance on human evaluators (e.g., Bob) presents another threat to external validity. One way to address the threat is to devise some objective way for evaluation, e.g., using random combination of uncommon words as a baseline. Finally, our particular selection of tools and parameters, such as the POS tagger and the threshold for determining dominant topics, may limit how the results could be generalized to other cases.

6 Discussion

So far, we have discussed how our combinational creativity framework can be applied to create new requirements for Firefox and Mylyn, and a study evaluating our framework by a professional software engineer (i.e., Bob). We have further discussed a second study involving humans manually performing combinational creativity, and compared the cost and effectiveness of our framework with those of the manual approach. In this section, we shed light on some observations and lessons learned throughout this research.

6.1 On our framework

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 used during the final phase (refer Table 5). Furthermore, we completely eliminate human intervention and automatically generate requirements from idea-pairs using linguistic templates. Note that human intensive activities to generate creative requirements can be time consuming and considerably exhausting, especially for someone who is not in her creative moments. Thereby, we consider the end-to-end automation to be the greatest strength of our framework. In addition, handling a large number of potential ideas, presented in any form, can be overwhelming for humans, whereas our framework does an excellent job of limiting the total number of likely possibilities.

One issue of our framework concerns the data sources of the input. When applying our framework to Firefox and Mylyn, we took advantage of the requirements and stakeholders' clarifications about the requirements stored in the project repositories. Using repository mining methods to detect the patterns of requirement clarifications can also lend support to project managers to assess the state of discussions around a requirement and promptly react to potential requirements problems [52]. The repository data, however, cannot be complete or even representative in terms of capturing stakeholders communications. Researchers have exploited other means such as e-mails [62] to build stakeholders social network in software engineering. As far as creativity is concerned, informal personal interactions, if leveraged, may provoke even more innovative ideas (topics) to be generated in our framework.

Another issue with our framework, also pointed out by Bob (refer Sect. 5), is that there is always a chance of creating some statements that may not make any sense to humans (e.g., $F5, F9, F10$, and $F11$ in Table 6). However, we observe that an apparently less meaningful statement can ignite creative thinking among humans as happened with Bob when he read the requirement $F4$. When asked about his feedback on our framework during the exit interview, one comment from Bob substantiates our observation.

> I do not think a less meaningful statement will always be completely useless.... Sometimes, paying a little more attention to it may provide an aha moment....

Expanding noun to a clause in multiple ways (refer Sect. 4) may seem to be excessive in some cases. We would like to emphasize at this point that this is an operationalization we follow for our demonstration. Our framework is flexible to support other linguistic approaches to formulate such a clause.

6.2 On innovative requirements

Requirements generated by our framework: After analyzing the ratings provided by Jim and Sara, we observe lower ratings ($\kappa = 0.71$) for some framework-generated requirements compared to the ratings our analyst provided (refer Sect. 5.1.2). However, the innovative aspects of these requirements are still perceived to be above neutral based on their average ratings. One reason behind these requirements receiving relatively lower rating is that they are presented in very short statements. Note that Jim and Sara were asked to rate the requirements as they appeared without making any modification. Given them an opportunity for further modification, we might have a different story for our automated requirements.

Requirements generated manually: Data presented in Table 8 indicate that considerable number of manually generated requirements obtained higher ratings from Jim and Sara. However, according to the information presented in Tables 9, 10, and 11, it is evident that most of the manually generated requirements are created using far more than two topic words and combining more than two topics. Our framework, on the other hand, combines exactly two topics. Such a flexible combination of topics during the manual process enabled the developers create reasonably elaborated requirements. We posit this aspect played an important role behind the ratings of the manually generated requirements. Furthermore, the participants performed combinational creativity using the topics generated through our framework. Therefore, the impact of the first three phases of our framework on the manually generated requirements cannot be ignored. A potential improvement, thus, is to combine automated and manual efforts toward interactive generation of creative requirements using mechanism's outputs.

6.3 On cost and effectiveness of our framework

Speed-wise, our automated framework generates creative requirements much faster than any manual approach (refer Sect. 5.2.6). While testing H1, we find clear statistical evidence at $\alpha = 0.05$ level of significance that our framework generates requirements far quickly than a manual method. We consider it to be one of the most powerful attributes of the framework, and we expect our framework to be faster than any manual method for combinational creativity.

While testing H2, we did not find enough statistical evidence of our framework generating higher-quality requirements than the manual approach at $\alpha = 0.05$ level of significance (refer Sect. 5.2.6. $t = 1.8, p = 0.11$). Note that our participants had the liberty of combining more than two topics, thereby generating more elaborated requirements. Considering these aspects, we conclude that the quality of the requirements generated by our framework is comparable to the manually generated requirements. Thus, our study shows evidence of the effectiveness of our framework.

6.4 On the identified creativity processes

In Sect. 5.2.2, we identified three different creativity process: Divide-and-Trade, Individual-Interleaved-with-Group, and Group-Heavy followed by three different groups of developers in elaborating requirements from topics. Results presented in Table 8 indicate further difference among these processes in terms of their performances. In what follows, we provide some additional insights along this line.

Interesting attributes of Divide-and-Trade: The uniqueness of this process is in its divide and conquer approach where each group member works with a subset of topics that leads to a better management and organization of the overall collaborative effort. In addition, using the concept of NFR indicators is a very interesting attribute of this approach.[5] These NFR indicators helped generate not only original requirements but also inspired organically generated ones.

On Individual-Interleaved-with-Group: Table 8 indicates that this process generates 13 requirements spending 300 min (maximum time spent) with just 34.62 % being *creative*. The reason lies in the way the process handles creativity activities. Recall that each participant worked individually with all the topics and generated their own requirements before the discussion phase (refer Fig. 12). As a result, each individual initially became overprotective about her requirement that delayed the final formulation and eventually led to a less efficient collaborative effort. The following comment by a participant demonstrates this fact.

> ....Sometimes we argued for a requirement we formulated before coming to the meeting. Although such conflicts were eventually resolved, it took some of our energy.

[5] The participants have not mentioned anything about NFR indicators. We identified this strategy while analyzing the requirements they generated.

We therefore believe automatically formulating creative requirements like our framework can be valuable for individual requirements engineers and their collaborations.

6.5 Limitations

The work presented in this paper contains the development and demonstration of a conceptual framework, as well as an exploratory study involving humans conducting combinational creativity activities. We discuss the limitations from both the framework and exploratory study-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. However, existing requirements and artifacts from a similar system, knowledge and ideas stored in online knowledge repositories, etc. can be used in such case, and our framework can be extended or modified accordingly.'

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 (refer Sect. 2.3), 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., [63]), instead of the social network, could be an alternative. We believe, however, considering the social network better reflects the collaborative nature of RE [30] and better supports the definition of combinational creativity [12].

The limitations of topic modeling and POS tagging [17, 18] are also relevant to our framework. However, neither of these techniques uses parsing, and the POS tagger can tolerate poor English. This is a strength of our framework in dealing with requirements and stakeholder communications as these descriptions may not follow perfect grammatical rules in practice. We also filter out unnecessary and most frequently used words before applying topic modeling. Therefore, we do not expect additional limitations when applying these techniques. The practical implementation of our framework revealed a small number of requirements for the subject systems. This is mainly due to a more conservative filtering criterion we applied during requirements generation using our framework (refer Sect. 4.2). Further relaxed approach during these activities should help identify a higher number of new requirements. Another limitation of our framework is that it does not modify the word itself to convert it into a traditional verb or noun. However, we believe the automatically generated requirements statements are complete enough for a human to ignite brainstorming and further understand the ideas conveyed by the statements.

In our opinion, the most important limitation of our framework is the possibility of generating some less-meaningful requirements at the end. Although more sophisticated linguistic templates can be used, we admit our framework will always have some limitation in this area. However, this limitation is not due to the construction of our framework per se, but because of the AI-complete problem of natural language understating in a mechanical manner [64]. We have demonstrated that the less meaningful statements may have the merit to ignite creative thinking among humans. Thus, we expect our framework to be useful to requirements engineers in general.

From the evaluation perspective: In order to compare the performance of our framework with manual approach, we conducted a study involving three groups of developers performing combinational creativity in RE. A more direct evaluation may involve comparing our POS-based combination of topics with some random combinations of unusual words. Such an evaluation may serve as a baseline to help contextualize to what degree our framework supports novelty. We recognize the limitation due to recruiting one expert during the first evaluation (see Sect. 5.1). However, the automated framework is built upon the positive evaluation results from 29 evaluators in our previous work [13]. Furthermore, the automated requirements are also evaluated by two more experts in the second study (see Sect. 5.2.6). Therefore, we posit that our findings are reliable.

In order to compare the creativity merit of the automated and manually generated requirements, we obtained qualitative ratings from two professionals (Jim and Sara). Whether conducting our study with additional groups or recruiting more experts to evaluate requirements would lead to a different conclusion is still an open question. In addition, our study focuses on humans conducting creativity activities in a collaborative setting. Exploring how solitary humans would perform require further studies. However, RE is inherently a highly collaborative process justifying the collaborative settings in our study. Furthermore, the average κ statistic showed a substantial agreement between the raters [50], thereby improving the reliability of our findings.

7 Conclusion

In this paper, we have contributed a novel framework that provides an end-to-end automated support for innovating requirements from a combinational creativity

perspective [2, 5]. A human subject evaluation shows promising practical implications of our framework. We further conduct a second study involving humans performing combinational creativity in RE, thereby investigating the cost and effectiveness of our framework with those of manual methods. Although there remain some limitations, the findings of these studies suggest that our framework generates creative requirements in a highly efficient manner. Our research further indicates the prospect of the mechanism's outputs in iterative creative requirements generation.

In the future, we plan to investigate the effectiveness of our framework against an objective baseline, such as random combinations of unusual words. We also plan to reduce our framework's dependency on existing requirements in order to expand its applicability to new software systems and live projects. Finally, we intend to push our research toward the dimension of transformational creativity in RE [12].

References

1. Lemos J, Alves C, Duboc L, Rodrigues GN (2012) A systematic mapping study on creativity in requirements engineering. In: Proceedings of the annual ACM symposium on applied computing (SAC), pp 1083–1088
2. Maiden N, Jones S, Karlsen K, Neill R, Zachos K, Milne A (2010) Requirements engineering as creative problem solving: a research agenda for idea finding. In: Proceedings of the international requirements engineering conference (RE), pp 57–66
3. Maiden N, Ncube C, Robertson S (2007) Can requirements be creative? Experiences with an enhanced air space management system. In: Proceedings of the international conference on software engineering (ICSE), pp 632–641
4. Sternberg RJ (1999) Handbook of creativity. Cambridge University Press, Cambridge
5. Boden MA (2003) The creative mind: Myths and mechanisms. Routledge, London
6. Maiden N, Gizikis A, Robertson S (2004) Provoking creativity: imagine what your requirements could be like. IEEE Softw 21(5):68–75
7. Maiden N, Manning S, Robertson S, Greenwood J (2004) Integrating creativity workshops into structured requirements processes. In: Proceedings of the ACM conference on designing interactive systems: processes, practices, methods, and techniques, pp 113–122
8. Maiden N, Robertson S (2005) Integrating creativity into requirements processes: experiences with an air traffic management system. In: Proceedings of the international requirements engineering conference (RE), pp 105–114
9. Karlsen IK, Maiden N, Kerne A (2009) Inventing requirements with creativity support tools. In: Requirements engineering: foundation for software quality. Springer, Berlin, pp 162–174
10. Zachos K, Maiden N (2008) Inventing requirements from software: an empirical investigation with web services. In: Proceedings of the international requirements engineering conference (RE), pp 145–154
11. Hariri N, Castro-Herrera C, Mirakhorli M, Cleland-Huang J, Mobasher B (2013) Supporting domain analysis through mining and recommending features from online product listings. IEEE Trans Softw Eng 39(12):1736–1752
12. Maiden N (2013) Requirements engineering as information search and idea discovery (keynote). In: Proceedings of the international requirements engineering conference (RE), pp 1–1
13. Bhowmik T, Niu N, Mahmoud A, Savolainen J (2014) Automated support for combinational creativity in requirements engineering. In: Proceedings of the international requirements engineering conference (RE), pp 243–252
14. Burt RS (2004) Structural holes and good ideas. Am J Sociol 110(2):349–399
15. Pirolli P (2009) An elementary social information foraging model. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI), pp 605–614
16. Linstead E, Lopes C, Baldi P (2008) An application of Latent Dirichlet Allocation to analyzing software evolution. In: Proceedings of the international conference on machine learning and applications (ICMLA), pp 813–818
17. Blei D, Ng A, Jordan M (2003) Latent Dirichlet Allocation. J Mach Learn Res 3:993–1022
18. Brill E (1992) A simple rule-based part of speech tagger. In: Proceedings of the workshop on speech and natural language, pp 112–116
19. Kersten M, Murphy G (2005) Mylar: a degree-of-interest model for ides. In: Proceedings of the international conference on aspect-oriented software development (AOSD), pp 159–168
20. Suwa M, Gero J, Purcell T (2000) Unexpected discoveries and S-invention of design requirements: important vehicles for a design process. Des Stud 21(6):539–567
21. Maher ML, Brady K, Fisher DH (2013) Computational models of surprise in evaluating creative design. In: Proceedings of the international conference on computational creativity (ICCC), pp 147–151
22. Ritchie G (2001) Assessing creativity. In: Proceedings of the AISB-01 symposium on AI and creativity in arts and science, pp 3–11
23. Nöbauer M, Seyff N, Maiden N, Zachos K (2011) S3c: using service discovery to support requirements elicitation in the erp domain. In: Proceedings of the international conference on advanced information systems engineering, pp 18–32
24. Lutz R, Patterson-Hine A, Nelson S, Frost CR, Tal D, Harris R (2007) Using obstacle analysis to identify contingency requirements on an unpiloted aerial vehicle. Requir Eng J 12(1):41–54
25. Salinesi C, Mazo R, Diaz D, Djebbi O (2010) Using integer constraint solving in reuse based requirements engineering. In: Proceedings of the international requirements engineering conference (RE), pp 243–251
26. Fuxman A, Pistore M, Mylopoulos J, Traverso P (2001) Model checking early requirements specifications in tropos. In: Proceedings of the international requirements engineering conference (RE), pp 174–181
27. Sakhnini V, Berry DM, Mich L (2010) Validation of the effectiveness of an optimized epmcreate as an aid for creative requirements elicitation. In: Proceedings of the 16th international working conference on requirements engineering: foundation for software quality, pp 91–105
28. Sakhnini V, Mich L, Berry DM (2012) The effectiveness of an optimized epmcreate as a creativity enhancement technique for web site requirements elicitation. Requir Eng 17(3):171–186
29. Lefons E, Pazienza M, Silvestri A, Tangorra F, Corfiati L, De Giacomo P (1977) An algebraic model for systems of psychically interacting subjects. In: Proceedings of the IFAC workshop information & systems, pp 155–163

30. Mahaux M, Nguyen L, Gotel O, Mich L, Mavin A, Schmid K (2013) Collaborative creativity in requirements engineering: analysis and practical advice. In: Proceedings of the international conference on research challenges in information science (RCIS), pp 1–10
31. Damian D, Marczak S, Kwan I (2007) Collaboration patterns and the impact of distance on awareness in requirements-centred social networks. In: Proceedings of the international requirements engineering conference (RE), pp 59–68
32. Begel A, Khoo Y, Zimmermann T (2010) Codebook: discovering and exploiting relationships in software repositories. In: Proceedings of the international conference on software engineering (ICSE), pp 125–134
33. Sarma A, Maccherone L, Wagstrom P, Herbsleb J (2009) Tesseract: interactive visual exploration of socio-technical relationships in software development. In: Proceedings of the international conference on software engineering (ICSE), pp 23–33
34. Wolf T, Schroter A, Damian D, Nguyen T (2009) Predicting build failures using social network analysis on developer communication. In: Proceedings of the international conference on software engineering (ICSE), pp 1–11
35. Asuncion H, Asuncion A, Taylor R (2010) Software traceability with topic modeling. In: Proceedings of the international conference on software engineering (ICSE), pp 95–104
36. Linstead E, Rigor P, Bajracharya S, Lopes C, Baldi P (2007) Mining concepts from code with probabilistic topic models. In: Proceedings of the international conference on automated software engineering (ASE), pp 461–464
37. Thomas S, Adams B, Hassan A, Blostein D (2010) Validating the use of topic models for software evolution. In: Proceedings of the IEEE working conference on source code analysis and manipulation (SCAM), pp 55–64
38. Porteous I, Newman D, Ihler A, Asuncion A, Smyth P, Welling M (2008) Fast collapsed gibbs sampling for Latent Dirichlet Allocation. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 569–577
39. Linstead E, Bajracharya S, Ngo T, Rigor P, Lopes C, Baldi P (2009) Sourcerer: mining and searching internet-scale software repositories. Data Mining Knowl Discov 18(2):300–336
40. Linstead E, Rigor P, Bajracharya SK, Lopes CV, Baldi P (2007) Mining internet-scale software repositories. In: Proceedings of the neural information processing systems (NIPS)
41. Wu J (2012) Advances in K-means clustering: a data mining thinking. Springer, Berlin
42. Borgatti SP, Everett MG, Freeman LC (2002) Ucinet for windows: software for social network analysis. Analytic Technologies
43. Bastian M, Heymann S, Jacomy M (2009) Gephi: an open source software for exploring and manipulating networks. In: Proceedings of the international conference on weblogs and social media (ICWSM), pp 361–362
44. Chang J, Boyd-Graber JL, Gerrish S, Wang C, Blei DM (2009) Reading tea leaves: how humans interpret topic models. In: Proceedings of the neural information processing systems (NIPS), vol 22, pp 288–296
45. Liaskos S, Lapouchnian A, Yu Y, Yu E, Mylopoulos J (2006) On goal-based variability acquisition and analysis. In: Proceedings of the international conference on requirements engineering (RE), pp 79–88
46. Niu N, Easterbrook S (2008) Extracting and modeling product line functional requirements. In: Proceedings of the international requirements engineering conference (RE), pp 155–164
47. Fillmore C (1968) The case for case. In: Bach E, Harms R (eds) Universals in linguistic theory. Holt, Rinehart and Winston, New York, pp 1–88
48. Hill E (2010) Integrating natural language and program structure information to improve software search and exploration. PhD. Thesis, University of Delaware
49. Gupta S, Malik S, Pollock L, Vijay-Shanker K (2013) Part-of-speech tagging of program identifiers for improved text-based software engineering tools. In: Proceedings of the international conference on program comprehension (ICPC), pp 3–12
50. Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval, vol 1. Cambridge University Press, Cambridge
51. Ernst NA, Murphy GC (2012) Case studies in just-in-time requirements analysis. In: IEEE international workshop on empirical requirements engineering, pp 25–32
52. Knauss E, Damian D, Poo-Caamano G, Cleland-Huang J (2012) Detecting and classifying patterns of requirements clarifications. In: Proceedings of the IEEE international requirements engineering conference (RE), pp 251–260
53. Liu H, Gao Y, Niu Z (2012) An initial study on refactoring tactics. In: Annual international computers, software & applications conference, pp 213–218
54. Scacchi W (2002) Understanding the requirements for developing open source software systems. IEE Softw 149(1):24–39
55. Niu N, Bhowmik T, Liu H, Niu Z (2014) Traceability-enabled refactoring for managing just-in-time requirements. In: Proceedings of the IEEE international requirements engineering conference (RE), pp 133–142
56. Zaman S, Adams B, Hassan AE (2011) Security versus performance bugs: a case study on firefox. In: Proceedings of the working conference on mining software repositories (MSR), pp 93–102
57. Griffiths TL, Steyvers M (2004) Finding scientific topics. In: Proceedings of the national academy of sciences of the United States of America, pp 5228–5235
58. Niu N, Savolainen J, Bhowmik T, Mahmoud A, Reddivari S (2012) A framework for examining topical locality in object-oriented software. In: Proceedings of the annual computer software and applications conference (COMPSAC), pp 219–224
59. Paulus PB, Nijstad BA (2003) Group creativity: innovation through collaboration. Oxford University Press, Oxford
60. Montgomery D, Runger G (2010) Applied statistics and probability for engineers. Wiley, New York
61. Yin RK (2008) Case study research: design and methods, vol 5. Sage, London
62. Bird C, Pattison D, D'Souza R, Filkov V, Devanbu P (2008) Latent social structure in open source projects. In: Proceedings of the ACM SIGSOFT international symposium on foundations of software engineering (SIGSOFT/FSE), pp 24–35
63. Niu N, Mahmoud A (2012) Enhancing candidate link generation for requirements tracing: the cluster hypothesis revisited. In: Proceedings of the international requirements engineering conference (RE), pp 81–90
64. Popescu A-M, Etzioni O, Kautz H (2003) Towards a theory of natural language interfaces to databases. In: Proceedings of the 8th international conference on intelligent user interfaces, pp 149–157