An Experimental Investigation of Reusable Requirements Retrieval

Anas Mahmoud
Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
amm560@msstate.edu

Nan Niu
Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
niu@cse.msstate.edu

Abstract

Building a requirements library allows practitioners to better leverage reuse since they can work on the abstractions closer to the software system's initial concepts. However, little research has explored how to retrieve requirements profiles to answer user's queries to the library. This paper presents an experimental investigation of retrieving functional requirements profiles. We assessed the effect of two factors: basic retrieval method and synonym support, and found that: 1) no significant differences in retrieval effectiveness were detected among the methods, but recall was significantly improved by having synonym support; 2) searching times were significantly different; and 3) different methods indeed retrieved different items.

1. Introduction

The effectiveness of software reuse can be evaluated in terms of *cognitive distance* – an intuitive gauge of the intellectual effort required to take a software system from one stage of development to another [5]. One way to reduce cognitive distance is for practitioners to work on the abstractions closer to the system's initial concepts, such as requirements specifications commonly written in natural language.

For requirements reuse to be attractive, a library is usually constructed to ensure that the overall reuse effort is less than the effort to elicit and document the requirements from scratch. In this context, reuse involves three steps: accessing, understanding, and adapting requirements. Thus, properly accessing (indexing and retrieving) a collection of requirements documents is central to reduce the effort of understanding and adaptation [9].

Much research to date has focused on indexing [4, 8, 9], i.e., generating requirements profiles to be stored in the library. Functional requirements profiles (FRPs) characterize a domain's action themes at a primitive level [8]. Each FRP models a single operational unit that yields an observable result of value to an external actor. While its classification scheme is detailed in the next section, FRP captures key function and service that must be implemented and delivered by a family of software systems [7].

Despite recent advances in indexing, there has been little empirical research into retrieving reusable requirements. Little is known about how to effectively and efficiently answer user's queries to the library. Such a shortfall hinders the ability to fully leverage reuse, since the main purpose of a profile is to play the role of requirement's surrogate during retrieval [4].

To shorten the gap, we conducted a study of retrieving FRPs. Our goal was to determine whether different retrieval methods would make a practically significant difference. To that end, the study reported here addressed the following research questions:

- Are any retrieval methods more effective than others for helping to find relevant FRPs?
- Are any retrieval methods more efficient in terms of searching time for finding FRPs?
- Do the methods retrieve the same FRPs?

The results of our investigation are particularly valuable for organizations that design and maintain library infrastructures, and for practitioners who search requirements information to be reused in new contexts.

2. Related work

2.1. Functional requirements profiles (FRPs)

Two approaches can be distinguished when constructing a profile for an artifact in the software repository: the free-text indexing approach as defined in information retrieval (IR), and the knowledge-based approach as defined in artificial intelligence (AI).

IR approach draws information from natural language documents. No semantic knowledge is used

Table 1. Classification scheme for FRPs

Agent	Action	Object	Process
operator	detect	incident	*<null>*
software-to-be	generate	response plan	automatically
operator	generate	response plan	manually

and no interpretation of the document is given. Maarek *et al.* used a two-word unit based on word co-occurrence to characterize manual pages and comments associated with the source code [6]. AbstFinder facilitated requirements elicitation by employing string-matching to find abstractions (word repetitions) among raw client information, e.g., interview transcripts [4]. IR approach relies solely on text as a source of knowledge, which is impracticable because clients cannot reduce all their demands to textual form [10]. Furthermore, IR-generated profiles do not necessarily have any meaning [4], making understanding and adapting them to achieve reuse extremely difficult.

In contrast, AI approach attempts to generate meaningful indices by taking semantic knowledge into account, which we prefer in our current work. Fry *et al.* analyzed source code comments and method identifiers via natural language processing, and found that verb-DO (direct object) pairs were useful in capturing action-oriented concerns [3]. In [9], the authors classified a code component via the scheme of functionality (what it does), environment (where it does it), and realization (how it does it). When coping with requirements, we shall move up from the level specifying *how* a function is implemented to the level describing the semantics of *what* is implemented [5].

Our earlier work introduced a semi-automatic technique to extract functional requirements profiles (FRPs) by supplementing expert knowledge with linguistic clues [8]. The foundation work [9] and our continuing investigation [7] led to, what we believe, an optimal set of attributes to describe the FRPs. Table 1 shows the classification scheme with examples of FRPs in the traffic management domain [7]. The four attributes: *agent* (actor), *action* (activity), *object* (theme), and *process* (means and manner), are chosen for a couple of reasons. First, they represent an underlying semantic case frame [7] of the domain's action themes. Second, algorithms and heuristics exist for reducing the manual effort in uncovering this knowledge structure from requirements artifacts [8].

Most research on reuse libraries lacks rigorous experimental evaluation of retrieval performance, e.g., [3] and [6] provided no statistical analysis to determine whether their methods were significant. A notable exception is the study of a UNIX shell library [2], which reported, among 4 representation methods, that:

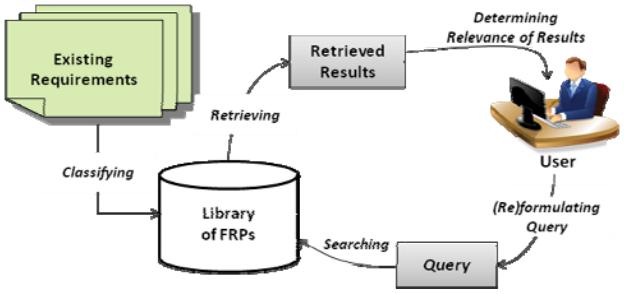


Figure 1. Library construction and access

- There were no significant differences in retrieval effectiveness.
- There were significant differences in searching times.
- The methods retrieved different items, with average pairwise overlaps for the methods ranged from 72% to 85%.

It is our intent in this paper to examine reusable *requirements* retrieval, though a single representation scheme is used in our experiment, as illustrated in Table 1.

2.2. Retrieval methods

Figure 1 shows that classifying requirements results in a library, which user searches iteratively for reusable components. We adopt the most commonly used search interface, a single textbox, so that user query can be flexibly expressed in an uncontrolled manner. A retrieval method ranks every F in the FRP-library according to the user query Q. Three basic retrieval methods are investigated in our work. For illustrative purposes, let query q_1 = "monitor incident" and two FRPs f_1 = (operator, monitor, incident, *<null>*), f_2 = (operator, relocate, incident, *<null>*).

– RM_1: Set, which treats F and Q as sets of words, and measures their similarity by Dice coefficient [11]:

$$S_1(F,Q) = \frac{2 \cdot |F \cap Q|}{(|F| + |Q|)}$$

Therefore, $S_1(f_1,q_1) = 4/6 = 0.67$, $S_1(f_2,q_1) = 2/6 = 0.33$.

– RM_2: Vector, which treats F and Q as vectors of terms over a bag (unordered collection) of words, and measures their similarity by the cosine of the angle between the vectors [11]:

$$S_2(F,Q) = \frac{\sum_{i=1}^{N} w_i \cdot q_i}{\sqrt{\sum_{i=1}^{N} w_i^2 \cdot \sum_{i=1}^{N} q_i^2}}$$

where N is the size of the vector, and w_i and q_i are term weights in F and Q respectively. We use tf-idf weights: $w_i = tf_i(f) \cdot idf_i$, $q_i = tf_i(q) \cdot idf_i$, where $tf_i(f)$ and $tf_i(q)$ are term frequency of $term_i$ in FRP f and user query q,

respectively. idf_i is the inverse document frequency, and is computed as $idf_i = \log_2(t/df_i)$, where t is the total number of FRPs in the library and df_i is the number of FRPs in which $term_i$ occurs. In our traffic management study, $S_2(f_1,q_1) = 0.86$, $S_2(f_2,q_1) = 0.24$.

– RM_3: Facet, which treats each F as a descriptor of ordered terms. Facets are sometimes considered as perspectives or dimensions of a particular domain [9]. Each facet is assigned a weight λ based on perceived importance. Our classification scheme shown in Table 1 gives rise to 4 facets, (*agent*, *action*, *object*, *process*), which we assign the normalized weights, (λ_1=0.15, λ_2=0.5, λ_3=0.25, λ_4=0.1), to highlight the action-oriented concerns encapsulated in verb-DO pairs [3]. The similarity is measured by matching every term of Q with every weighted-facet term of F:

$$S_3(F,Q) = \sum_{i=1}^{N}\sum_{j=1}^{M}\sigma(f_i,q_j) \text{ where } \sigma(f_i,q_j) = \begin{cases} \lambda_i & \text{if } f_i = q_j \\ 0 & \text{otherwise} \end{cases}$$

Therefore, $S_3(f_1,q_1) = 0.5+0.25 = 0.75$, $S_3(f_2,q_1) = 0.25$.

To extend the above basic retrieval methods to cope with close but inexact matches, a similarity coefficient $\sigma \in [0,1]$ is used to match synonyms. In this way, we are able to accommodate conceptual closeness [9] by integrating the main part of S_1, S_2, and S_3 with σ. We use S'_i to denote the synonym-augmented similarity measure of S_i, and RM'_i to denote the synonym-augmented counterpart of RM_i, where i=(1,2,3). For example, if ("monitor", "relocate") is a synonym pair with σ=0.75, then $S'_1(f_2,q_1) = [2 \cdot (0.75+1)] / 6 = 0.58$, $S'_2(f_2,q_1) = 0.49$, and $S'_3(f_2,q_1) = (0.75 \cdot 0.5)+0.25 = 0.63$. In all cases, synonym-enabled matching outperforms synonym-disabled one, but underperforms exact matching. In other words, $S_i(f_2,q_1) < S'_i(f_2,q_1) < S_i(f_1,q_1)$ with i=(1,2,3). These results demonstrate the enhancement and correctness of the synonym-enabled retrieval methods.

3. Experimental design

3.1. Hypotheses

To address our research questions, the null hypothesis was formulated as:

- H_0. There is no significant difference among the methods in terms of retrieval effectiveness, retrieval efficiency, and retrieved items.

The alternative hypotheses were then stated as:

- H_1. Some retrieval methods are significantly more effective than others.
- H_2. Some retrieval methods are significantly more efficient than others.
- H_3. Different methods retrieve different items.

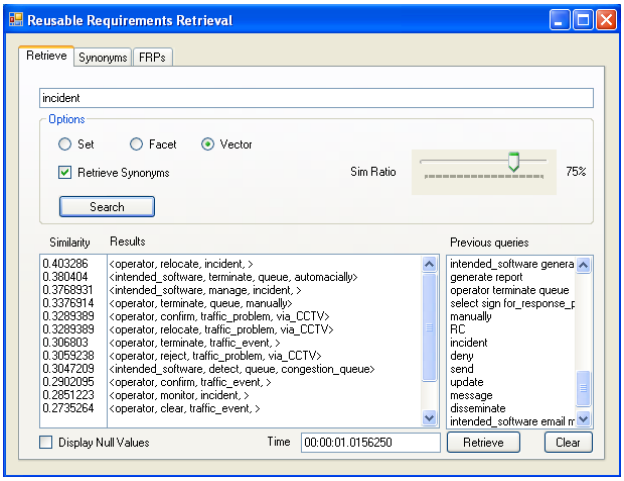


Figure 2. Screenshot of the prototype retrieval tool

3.2. Instrumentation

We fully implemented the retrieval methods described in Section 2.2. Figure 2 shows a screenshot of our prototype tool written in C#. The top textbox accepts the user query in a case insensitive way. The control panel is used to specify the search settings: which retrieval method to use (radio button), whether synonym augmentation is enabled (checkbox), and what synonym similarity coefficient to use (slider). The retrieved results, together with similarity ratios, are displayed in the list box. We also record the time spent in retrieval, and keep track of the query history to help the users to quickly access the previous results and progressively reformulate their queries.

Our tool also provides the interfaces (tags) to manage the library, e.g., to add, edit, or remove FRPs and synonym classes. In this study, we constructed a library of FRPs in the traffic management domain. A traffic management system is designed mainly for detecting road incidents and disseminating the information to related parties. We collected four traffic management software requirements specifications (SRS's) from a multi-disciplinary organization offering services in many areas of practice, including transportation and systems [7].

The practitioners of the partner company followed the IEEE-STD-830 standard to structure the 4 related but distinct SRS's, making it easier to extract synonym classes and FRPs. We leveraged the previous FRP-extraction support [8] and manually validated the results. We identified 26 synonym classes and 101 FRPs. The size of our collection (101) is comparable to that reported in [2] and [6], and thus provides good generalizability of the results.

We adopted the procedure described in [6] to devise the queries. In our context, a typical query would be

expressed by a list of open-class words including nouns, verbs, adjectives, and adverbs. Our experimental design included 30 queries over the vocabulary of traffic management SRS's. This number (30) is comparable to that (28) in [2]. Our experiment used a repeated measures factorial design [12]. The appeal of this design is that it increases the test's statistical power for a given number of queries.

Each query in the test collection was then searched through our tool (cf. Figure 2). In order to compare the retrieved items from different methods, a cutoff must be determined [6]. In our experiment, the cutoff value (retrieval threshold) was set to 60%, according to the experience reported in [1]. This addresses a practical concern in that the user is unlikely to examine a long list of FRPs for reuse. The top 60% of the retrieval results were judged for relevance by experts.

3.3. Variables

– **Independent Variables**: 1) METHOD, which has 3 values: Set (RM_1, RM'_1), Vector (RM_2, RM'_2), and Facet (RM_3, RM'_3); 2) SYNONYM, which has 2 values: disabled (RM_1, RM_2, RM_3) and enabled (RM'_1, RM'_2, RM'_3).

– **Controlled Variables**: The library was built based on the FRPs extracted from industry SRS's. The search was performed through our tool. The synonym similarity coefficient (σ) was set to 0.75 for all queries.

– **Dependent Variables**: The dependent variables were grouped in three categories. (a) Effectiveness was assessed by well-known IR metrics, *precision*, *recall*, and *F_1-measure* [11]. Precision (*P*) measures accuracy and is defined as the proportion of retrieved information which is relevant. Recall (*R*) measures coverage and is defined as the proportion of retrieved relevant information to the total amount of all relevant information. F_1-measure is the harmonic mean of *P* and *R*: $F_1 = 2 \cdot P \cdot R / (P+R)$. (b) Efficiency was assessed in terms of searching time. (c) Overlap was assessed pairwise [2]: $X = |m_i \cap m_j| / |m_i \cup m_j|$, where m_i and m_j are the FRPs retrieved by methods RM_i and RM_j.

4. Results

This section presents the data collected during the experiment and our quantitative data analysis. We reported descriptive statistics in boxplot and inferential statistics via ANOVA (analysis of variance) [12]. A boxplot reveals much about the data: its dispersion, its center, and how skewed the data is. Side-by-side boxplots quickly illustrate the relationships of these characteristics for multiple data distributions. ANOVA is a collection of statistical models and procedures, in which the observed variance is partitioned into components due to different explanatory variables. We used the 0.05 alpha level (α=0.05) to test the significance of difference among retrieval methods.

4.1. Retrieval effectiveness – H_1

The effectiveness, as measured by precision, recall, and F_1-measure, is the most important performance criterion [6]. It refers to the method's ability to provide relevant reusable information as needed by the user.

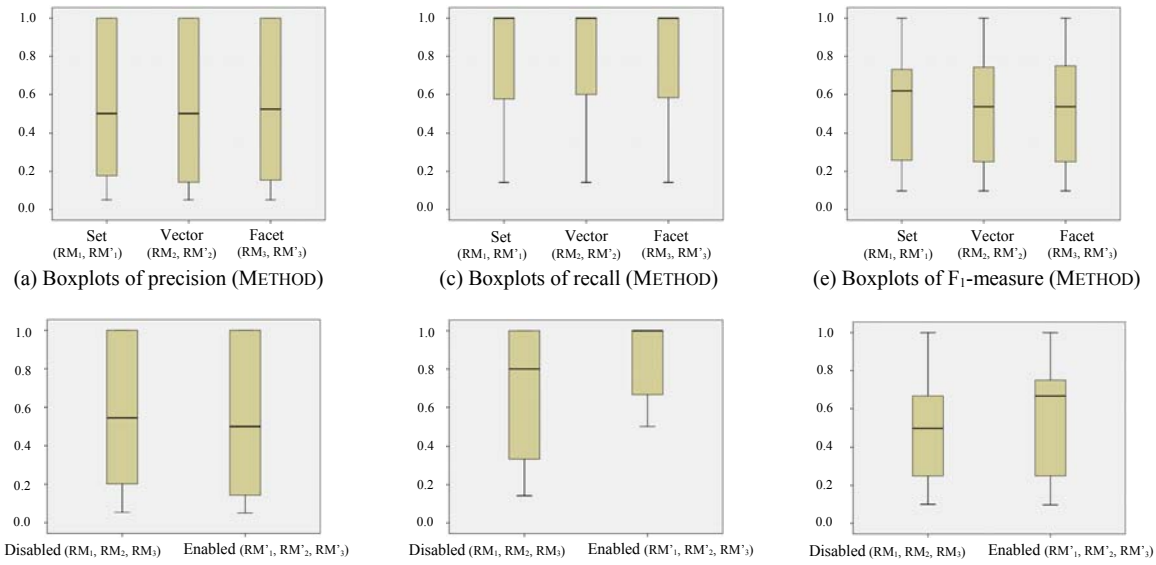
Precision. The boxplots in Figures 3(a) and 3(b) show the precision performance of the retrieval methods. The plots show that precision values varied between 0.05 and 1, with median values very close to each other. The two-way ANOVA results for precision, shown in Table 2(a), indicate that there is no significant interaction between METHOD and SYNONYM (F=1.727, p=0.208). The ANOVA results also show that neither independent variable has significant effect on precision: METHOD (F=0.026, p=0.956), SYNONYM (F=0.024, p=0.879).

Recall. The boxplots in Figure 3(c) show that recall values for METHOD is almost the same, whereas the boxplots in Figure 3(d) indicate a significant recall difference in terms of whether SYNONYM is supported. The two-way ANOVA results, shown in Table 2(b), reveal that the interaction of METHOD and SYNONYM has no significant effect on the recall values (F=1.649, p=0.212). The analysis also reveals that METHOD has no significant effect on recall (F=0.028, p=0.934). However, a significant effect of SYNONYM on recall is detected (F=14.632, p=0.002).

F_1-measure. The boxplots in Figures 3(e) and 3(f) show that the F_1-measure varies between 0.1 to 1, with median values varying from about 0.5 to 0.7. The two-way ANOVA results for F_1-measure, shown in Table 2(c), are similar to the results for recall. The interaction of METHOD and SYNONYM has no significant effect on the F_1-measure (F=2.572, p=0.108). There is no significant difference of the METHOD used (F=0.074, p=0.868); however, a significant difference exists when SYNONYM is supported (F=6.116, p=0.027).

4.2. Retrieval efficiency – H_2

The boxplots in Figure 4 show searching times in seconds. The searching time was calculated as the total time our prototype tool took to process the user query, access the FRPs, compute their similarities, and rank and display the final results. The searches were performed on a PC with a 1.80GHz P-4 CPU and 2GB RAM. Table 3 shows the results of a repeated-measure two-way ANOVA test for retrieval efficiency.



(a) Boxplots of precision (METHOD) (c) Boxplots of recall (METHOD) (e) Boxplots of F_1-measure (METHOD)

(b) Boxplots of precision (SYNONYM) (d) Boxplots of recall (SYNONYM) (f) Boxplots of F_1-measure (SYNONYM)

Figure 3. Descriptive statistics for retrieval effectiveness

Table 2. Inferential statistics for retrieval effectiveness

(a) ANOVA results for precision

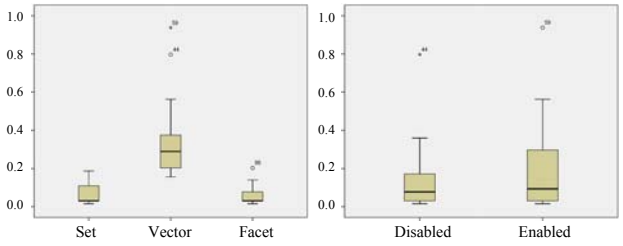
Source	Mean Square	df	F	Sig
METHOD	0	1.631	0.026	0.956
SYNONYM	0	1	0.024	0.879
METHOD * SYNONYM	0.002	1.263	1.727	0.208

(b) ANOVA results for recall

Source	Mean Square	df	F	Sig
METHOD	0.001	1.428	0.028	0.934
SYNONYM	0.937	1	14.632	0.002
METHOD * SYNONYM	0.002	1.902	1.649	0.212

(c) ANOVA results for F_1-measure

Source	Mean Square	df	F	Sig
METHOD	0	1.41	0.074	0.868
SYNONYM	0.35	1	6.116	0.027
METHOD * SYNONYM	0.003	1.608	2.572	0.108



(a) METHOD (b) SYNONYM

Figure 4. Boxplots of searching time

Table 3. ANOVA results for searching time

Source	Mean Square	df	F	Sig
METHOD	1.460	1.001	47.022	.000
SYNONYM	.058	1.000	81.895	.000
METHOD * SYNONYM	.066	1.220	94.822	.000

Our analysis shows that, in terms of searching time, there is a significant difference of the METHOD used, of whether SYNONYM is supported, and of the interaction between METHOD and SYNONYM. The vector model (RM_2, RM'_2) was the slowest in that the time complexity of tf-idf was quadratic $\mathcal{O}(t^2)$, while the set and facet models were linear $\mathcal{O}(t)$ in our implementation, where t is the total number of FRPs stored in the library.

Table 4. Pairwise overlaps for retrieval methods

	RM'_1	RM_2	RM'_2	RM_3	RM'_3
RM_1	.65	.69	.49	.75	.55
RM'_1		.55	.63	.57	.80
RM_2			.54	.70	.49
RM'_2				.46	.62
RM_3					.62

4.3. Retrieval overlap – H_3

Table 4 shows the retrieval overlap analysis results. For each pair of methods, the overlap was calculated for each query by computing the ratio of the number of FRPs in the intersection of the methods divided by the number of FRPs in their union. Then, the ratios from all queries were averaged to obtain one final overlap value. The overlap measures in Table 4 range from 46% to 80%. Note that these overlaps are much lower than those reported in [2]. A repeated-measure ANOVA implies that different method pairs do indeed retrieve significantly different items (F=3.991, p=0.016).

- The final results can be summarized as follows.
  - We failed to reject the null hypothesis in terms of retrieval effectiveness for METHOD used, but recall, a measure of coverage, was significantly improved by having SYNONYM support.

- We rejected the null hypothesis in terms of retrieval efficiency, i.e., H_2 could be accepted.
- We rejected the null hypothesis in terms of retrieved items, i.e., H_3 could be accepted.

5. Threats to validity

Several factors can affect the validity of our study. *Construct validity* is the degree to which the variables accurately measure the concepts they purport to measure [12]. We used three retrieval methods and their representative similarity measures: Set (Dice coefficient), Vector (tf-idf), and Facet (weighted attributes). Set retrieval uses object values; Vector uses object scalars; Facet leverages semantic knowledge. Although enhanced similarity measures (e.g., LSI [11] for Vector) and methods (e.g., ontology search) can be further experimented, we feel our chosen methods have covered the basic retrieval mechanisms. Synonym support is crucial to retrieving textual requirements; we therefore managed to measure the effect of SYNONYM explicitly. As for dependent variables: effectiveness, efficiency, and overlap, we do not feel that following their well-defined operational measures [2] posed a serious limitation.

Regarding *internal validity* [12], a limitation might be the results' tight coupling with our retrieval tool's particular implementation. As noted in [2], however, it is experimentally impossible to separate the method from the interface embedded in a tool. We addressed the threat by validating our tool on the FRPs extracted from student projects' requirements [8]. Another likely confounding variable is the test queries used in our experiment. This threat was mitigated by devising a comparable number of queries and by performing repeated measures on each of the queries.

We used industrial requirements to construct a library of substantial size. This helped address *external validity* [12] and thus allowed our results to be generalized to other application domains than transportation. Finally, in terms of *conclusion validity* or *reliability* [12], we expect that replications of our study should offer results similar to ours.

6. Conclusions

Constructing a library of requirements profiles reduces cognitive distance [5] for practitioners to leverage reuse. However, little is known about how to retrieve requirements profiles to answer user's queries to the library. This paper reports an experimental investigation of retrieving functional requirements profiles (FRPs). We found that: 1) no significant differences in retrieval effectiveness existed between the three basic methods, but recall was significantly improved by having synonym support; 2) searching times were significantly different; and 3) different methods retrieved different items, with average pairwise overlaps ranged from 46% to 80%. Our results were in line with the findings of a code library experiment [2]. The main difference was our focus on *requirements* retrieval (e.g., synonym support) rather than code retrieval; the overlaps in our experiment were much lower than those in [2]. In addition, we contributed a prototype tool to help practitioners construct, maintain, and search a library of FRPs.

The most important future work is to replicate our study with larger libraries and more retrieval methods. Such replications are necessary to strengthen the validity of the findings reported here. It would also be valuable to include human subjects, e.g., indexers and reusers, in the experimental design to investigate the requirements understanding and adapting problems.

Acknowledgments. *We thank the Empirical Software Engineering research group at Mississippi State University for helpful discussions. We especially thank Dr. Edward B. Allen and Dr. Gary L. Bradshaw for insightful comments.*

References

[1] H.M. Avila and J. Hüllen, "Feature Weighting by Explaining Case-Based Planning Episodes", *Proc. European Workshop on Advances in Case-Based Reasoning*, pp. 280-294, 1996.

[2] W.B. Frakes and T.P. Pole, "An Empirical Study of Representation Methods for Reusable Software Components", *IEEE Transactions on Software Engineering*, 20(8): 617-630, 1994.

[3] Z.P. Fry, D. Shepherd, E. Hill, L. Pollock, and K. Vijay-Shanker, "Analysing source code: looking for useful verb-direct object pairs in all the right places", *IET Software*, 2(1): 27-36, 2008.

[4] L. Goldin and D.M. Berry, "AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements", *Automated Software Engineering*, 4(4): 375-412, 1997.

[5] C.W. Krueger, "Software Reuse", *ACM Computing Surveys*, 24(2): 131-183, 1992.

[6] Y.S. Maarek, D.M. Berry, and G.E. Kaiser, "An Information Retrieval Approach For Automatically Constructing Software Libraries", *IEEE Transactions on Software Engineering*, 17(8): 800-813, 1991.

[7] N. Niu, *Extractive Product Line Requirements Engineering*, Ph.D. thesis, Univ. of Toronto, 2009.

[8] N. Niu and S. Easterbrook, "Extracting and Modeling Product Line Functional Requirements", *Proc. IEEE International Requirements Engineering Conference*, pp. 155-164, 2008.

[9] R. Prieto-Diaz and P. Freeman, "Classifying Software for Reusability", *IEEE Software*, 4(1): 6-16, 1987.

[10] K. Ryan, "The Role of Natural Language in Requirements Engineering", *Proc. IEEE International Symposium on Requirements Engineering*, pp. 240-242, 1993.

[11] Salton, G. and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[12] Wohlin, C., P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publishers, 2000.