Automatically Tracing Dependability Requirements via Term-Based Relevance Feedback

Wentao Wang, *Student Member, IEEE*, Arushi Gupta, Nan Niu, *Senior Member, IEEE*, Li Da Xu, *Fellow, IEEE*, Jing-Ru C. Cheng, and Zhendong Niu

***Abstract*—In many critical industrial information systems, tracking a dependability requirement is instrumental to the verification and validation (V&V) of security, privacy, and other dependability concerns. Automated traceability tools employ information retrieval methods to recover candidate links, which saves much manual effort. Integrating relevance feedback (RF) could potentially improve the retrieval effectiveness by soliciting the relevance judgments on a subset of the retrieval results and then incorporating the feedback into subsequent retrieval. However, little is known about how to use RF to trace dependability requirements. In this paper, we propose a novel term-based RF algorithm that leverages the term usage context to recommend positive and negative feedback. Experiments on two software datasets show that our algorithm significantly outperforms the contemporary link-based RF tracing method. Our work not only contributes a new solution to dependability requirements' V&V, but also enables further automation to reduce the manual effort in the development life cycle of dependable industrial systems.**

***Index Terms*—Dependability, dependability requirements, privacy, requirements tracing, relevance feedback (RF), security.**

I. Introduction

DEPENDABILITY is a critical quality attribute of many industrial systems such as medical applications [1], controller area networks [2], and power platforms [3]. Dependability refers to the system's ability to deliver service that can justifiably be trusted, and for different systems, dependability concerns range from safety through reliability to availability [4]. Because hardware redesigning is expensive, software plays an increasingly important role in continuously securing dependability in industrial informatics.

In software and systems engineering, *traceability* refers to the potential to connect interrelated artifacts throughout the life cycle of a system. To be able to trace various dependability concerns, such concerns must exist in the first place. Identifying those concerns, therefore, receives much attention in dependability requirements engineering [5].

In contrast, little effort has been devoted to developing effective tracing methods that facilitate the verification and validation (V&V) of dependability requirements. In other words, even if all the important requirements are identified, system dependability cannot be assured without the proper implementation of those requirements. Verification, in this context, is to ensure that the implementation conforms to the dependability specifications, and validation is to ensure that the system meets the stakeholders' expectations on dependability.

To reduce the manual effort, researchers have exploited information retrieval (IR) to automate the requirements tracing for V&V [6]–[8]. Each requirement's textual description serves as a query, against which the source code elements are ranked in the order of estimated relevance. However, IR-based tracing methods return a large portion of false positives, partly due to the many yet superfluous terms appeared in the query requirement where a term is a content identifier typically encapsulated in a word [9]. The false positives not only negatively affect the effectiveness of the tool support, but also decrease the trust from the engineers.

To increase the automated tracing tool's believability, Hayes *et al.* [6] suggested to solicit analyst feedback and incorporate it into the regeneration of candidate links such that the final traces could become as accurate as possible. The mechanism is known as *relevance feedback* (RF) and has received considerable attention in the IR literature [10].

The goal of our work is to explore ways that best leverage RF to trace dependability requirements. To that end, this paper makes four main contributions: developing a novel term-based RF algorithm to augment IR-based tracing, devising domain-specific dependability taxonomies to inform V&V, define a new

Manuscript received January 10, 2016; revised May 9, 2016, June 30, 2016, and October 13, 2016; accepted December 5, 2016. Date of publication December 7, 2016; date of current version January 3, 2018. The work was supported in part by the U.S. National Science Foundation (Award CCF 1350487) and in part by the National Natural Science Foundation of China (Fund No. 61375053). Paper no TII-16-0025. *(Corresponding author: N. Niu.)*

W. Wang, A. Gupta, and N. Niu are with the Department of Electrical Engineering and Computing Systems, University of Cincinnati, Cincinnati, OH 45221 USA (e-mail: wang2wt@mail.uc.edu; gupta2ai@mail.uc.edu; nan.niu@uc.edu).

L. D. Xu is with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: bjtuxu@gmail.com).

J.-R. C. Cheng is with the Information Technology Laboratory, U.S. Army Engineer Research and Development Center, Vicksburg, MS 39180 USA (e-mail: ruth.c.cheng@usace.army.mil).

Z. Niu is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: zniu@bit.edu.cn).

Digital Object Identifier 10.1109/TII.2016.2637166

metric to measure tracing method's specificity, and performing experimental evaluations of our proposed method.

II. Background and Related Work

A. Engineering Dependability Requirements

As industrial information and control systems become pervasive in our daily lives, we have a natural dependence on the proper construction and operation of these systems. Dependable systems are those with the capability of avoiding severe and/or frequent service failures [4]. The past decade has seen considerable advances in technology for building dependable industrial applications [1]–[3].

An emerging trend in dependable industrial engineering is the increasing exploitation of the fast evolution pace of software in the entire development life cycle [11], and preferably at the requirements level [12], [13]. So far, the research focus of security requirements engineering has been on elicitation [5]. While this is important, ensuring the requirements are implemented in the code base is equally important. In fact, certain dependability requirements are readily available, well documented, and therefore, must be rigorously satisfied. Examples include the Health Insurance Portability and Accountability Act (HIPAA)[1] which all healthcare-related products in the USA must comply with, as well as the IEC 61508 functional safety standard[2] that critical environments like automotive or energy production applications should follow.

In sum, dependability is a concern that must be taken into consideration starting from the early stages of industrial system development. Dependability requirements engineering has resulted in many approaches to eliciting specific concerns, most notably security and privacy. Less effort is made to ensure the realization of dependability requirements across the development life cycle. Next, we review the literature on automated traceability as a way to achieve requirements V&V.

B. Requirements Traceability and RF

The traceability information is instrumental in V&V to ensure that the right processes have been used to build the right system [6]. Tracing based on IR aims at automatically identifying candidate links between different types of software artifacts by relying on the artifacts' textual descriptions. Researchers have applied many IR methods in automated tracing. In most cases, a recall of 90% is achievable at precision levels of 5%–30%. In traceability, recall measures the percentage of true links found by IR algorithms, and precision measures the accuracy of the returned candidate link list [6]:

$$\text{Recall} = \frac{\sum_{q\in Q} r_q}{\sum_{q\in Q} R_q}, \quad \text{Precision} = \frac{\sum_{q\in Q} r_q}{\sum_{q\in Q} n_q} \tag{1}$$

where each $q \in Q$ is a to-be-traced requirement, R_q is the set of true links of q, and n_q is the set of candidate links that the IR method returns, out of which r_q are true. Because achieving high precision and high recall is a balancing act, the weighted harmonic mean of F_2 is commonly used as a single metric to measure tracing effectiveness [6]:

$$F_2 = \frac{5 \cdot \text{Precision} \cdot \text{Recall}}{4 \cdot \text{Precision} + \text{Recall}}. \tag{2}$$

To improve the retrieval effectiveness and hence the tracing tool's believability, Hayes *et al.* [6] proposed to integrate RF into the tracing process. In traditional IR, the basic idea of RF is to present an initial set of retrieved documents to the user and ask her to judge which documents are relevant to her information needs. The relevance judgments are used to produce a modified version of the query by weighting more on the terms appeared in the relevant documents and less on the terms from the irrelevant ones. The modified query is then used to retrieve a new set of documents [10].

Developed using the vector space model, the standard Rocchio method [14] remains an effective and robust RF mechanism [15]. Specifically, Rocchio's formula for modifying the query vector $\overrightarrow{Q_m}$ from the original query vector $\overrightarrow{Q_o}$ is:

$$\overrightarrow{Q_m} = (\alpha \cdot \overrightarrow{Q_o}) + \left(\beta \cdot \frac{1}{|D_r|} \sum_{\vec{d_j} \in D_r} \vec{d_j}\right) - \left(\gamma \cdot \frac{1}{|D_i|} \sum_{\vec{d_k} \in D_i} \vec{d_k}\right) \tag{3}$$

where D_r is the set of relevant documents and D_i is the set of irrelevant documents. Intuitively, the modified query takes the initial query vector while adding the weighted vectors $\vec{d_j} \in D_r$ and subtracting the weighted vectors $\vec{d_k} \in D_i$. Weighting parameters α, β, and γ are used to assign different emphases to $\overrightarrow{Q_o}$, positive feedback, and negative feedback, respectively.

Hayes *et al.* [6] provided evidence for standard Rocchio's accuracy improvement in IR-based tracing. Using simulated RF, other researchers were also able to show Rocchio's positive effect on tracing performances. In particular, the recent work by Panichella *et al.* [8] presented an adaptive version of RF. Rather than applying the standard Rocchio algorithm to every pair of tracing source and target, the adaptive Rocchio algorithm checks certain conditions before applying the RF. The conditions are defined based on the software artifacts' verbosity and the links already classified.

In sum, IR-based tracing algorithms reduce much manual effort in checking the implementation of requirements in the software development life cycle. When RF is integrated, the tracing effectiveness can be improved. A gap is to perform RF on a subset of the terms used to express specific dependability concerns, rather than adjusting weights for all the terms appeared in the query requirement. Next, we present a term-based RF algorithm for dependability requirements tracing.

III. Term-Based RF for Automated Tracing

It is important to point out here that, as far as the V&V of dependability requirements is concerned, tracing is *a* means, but *not the* means. Software testing represents another way to achieve V&V [11]; however, testing requires the software to be fully compilable, and at least partially, executable. Tracing,

[1] http://www.hhs.gov/hipaa/
[2] http://www.iec.ch/functionalsafety/

Input: dependability requirement $\vec{r}$, source code C
Output: ranked list of candidate traceability links for $\vec{r}$

Main Procedure
1. β_RF, γ_RF, and α_RF $\leftarrow$ **RF_Terms**($\vec{r}$, C)
2. **For** each term $t \in \alpha$_RF
3. $\quad \vec{r}_{t_{new}} = \alpha \cdot \vec{r}_{t_{old}}$
4. **For** each term $t \in \beta$_RF
5. $\quad \vec{r}_{t_{new}} = (\alpha + \beta) \cdot \vec{r}_{t_{old}}$
6. **For** each term $t \in \gamma$_RF
7. $\quad \vec{r}_{t_{new}} = (\alpha - \gamma) \cdot \vec{r}_{t_{old}}$
8. Generate ranked list of links

RF_Terms($\vec{r}$, C)
9. **Initialization** β_RF $=$ γ_RF $=$ u_RF $= \phi$
10. $T_r \leftarrow$ 5 tf-idf top-ranked terms from r **such that** the term appears in C
11. **For** each t $\in T_r$
12. nGram$_{t,n} \leftarrow$ n-grams from C containing t **where** n=$[2, 6]$
13. Reg$_{t,n} \leftarrow$ **Regular_Use**(nGram$_{t,n}$)
14. **If** Reg$_{t,n}$=ϕ for more than half of t $\in T_r$
15. //do nothing, meaning that r needs no RF
16. **Else**
17. **For** each gram $\in$ Reg$_{t,n}$
18. **If** [$\exists$ (gram $-$ t)] and t co-occur in the same sentence of r
19. β_RF $\leftarrow$ β_RF $\cup$ gram
20. **Elseif** [$\forall$ (gram $-$ t)] and t do not co-occur in the same sentence of r
21. γ_RF $\leftarrow$ γ_RF $\cup$ gram
22. **Else**
23. α_RF $\leftarrow$ α_RF $\cup$ gram
24. **Return** β_RF, γ_RF, u_RF by removing Java keywords

Regular_Use(nGram$_{t,n}$)
25. Reg$_{t,n} \leftarrow \phi$
26. **For** i = 2 to 6
27. **If** nGram$_{t,i}$ fits power law distribution
28. Reg$_{t,n} \leftarrow$ Reg$_{t,n} \cup$ Head_of_nGram$_{t,i}$
29. **Return** Reg$_{t,n}$

Fig. 1. Tracing dependability requirements via term-based RF.

therefore, complements methods like testing in that only textual information of the software artifacts is exploited.

Our analysis of dependability requirements in the experimental datasets shows that these requirements contain specific terms indicating concerns like security and privacy. This observation is consistent with the dependability regulations such as HIPAA and IEC 61508 mentioned above. When RF is applied, it is this specific set of dependability terms, rather than all the terms in a candidate traceability link, whose weights should be adjusted. Note that the need for finer-grained control over the terms was recognized by Shin and Cleland-Huang [16], but their approach was manual. We, therefore, contribute in Fig. 1 an automated algorithm that integrates term-based RF in dependability requirements tracing.

We adopt the n-gram models [17] to capture the statistical regularity in the code (see Lines 25−29 of Fig. 1). We consider 2 to 6 g to balance term usage context with noisy information [17]. Unlike [17], if two n-grams have exactly the same terms but different orderings, we consider them as one n-gram and increase its frequency of occurrence accordingly. The frequency of occurrence of the n-grams of a given n is either flat or follows a power-law-like distribution [18]. In the latter case, we take the most appeared n-grams (i.e., head of long tail) as regularities.

The n-gram analysis is performed only on the five most important terms for each dependability requirement (see Line 10 of Fig. 1). Our main rationale is that RF is effective in the vector space IR model if the weights of a few dimensions (terms), rather than all the dimensions, are adjusted [15]. On average, 2.3 out of 5 top terms represent dependability concerns. For example, in a security-critical requirement of iTrust[3] (namely, Use Case 2), the top five terms are "admin," "password," "secret," "database," and "agent." The first three terms are strongly dependability oriented, whereas the latter two are relatively general.

In our algorithm, if a resulting term's regular usage is empty, no RF will be further defined on top of it. This case shows that the term has high inverse document frequency value, and therefore, its appearance is relatively concentrated. An example is "electronic" that appears in only one requirement of iTrust (Use Case 3). Adjusting the weight of such terms has little effect on improving retrieved results. For the same reason, if more than half of a requirement's terms have no regular use in the source code, we do not think RF should be applied to that requirement (see Lines 14−15 of Fig. 1).

For the remaining regular n-grams, we categorize them in Lines 17−23 of Fig. 1 with three groups: α_RF, β_RF, and γ_RF. If any term of the regular n-gram and the key term co-occur in the same requirements sentence, we think both the sentence and the source code where the n-gram occurs describe the same concept. As a result, such an n-grams terms' weights should be increased. For example, one regular n-gram "change session time out" is deemed positive feedback, since "session" is one of the top five terms from iTrust's Use Case 3 and the term "out" co-occurs with "session" in one of the sentences of Use Case 3: "An authenticated session ends when the user logs out or closes the iTrust application."

In contrast, although "view" is a top five term of iTrust's Use Case 21 and "view patient office visit history" is regular in the code, none of the terms from this n-gram except for "view" appear in the requirement at all. This poses a strong signal that "view" should be weighted less when tracing this iTrust requirement, because the regular code usage pattern of "view" bears little relationship to the requirement. Finally, if a term belongs to neither positive (β_RF) nor negative (γ_RF), then it falls into α_RF and its weight is kept intact.

IV. Experimental Evaluation

A. Datasets and Measures

Two datasets are used to conduct the experiments in this paper. Table I shows the traceability-related characteristics of the datasets. Both projects are developed in Java and the correct trace links are defined by projects' original developers. Tables II and III list the dependability requirements.

The classification of dependability requirements in both projects is currently performed manually. We use the concepts presented in [4] as guidelines. Take iTrust's Use Case 1 ("create and disable patients") as an example, the requirement explicitly states that: "The HCP (health care professional) does

[3] http://agile.csc.ncsu.edu/iTrust

TABLE I
TRACEABILITY-RELATED CHARACTERISTICS OF DEPENDABILITY REQUIREMENTS (DEP.) AND NON-DEPENDABILITY REQUIREMENTS (NON-D.) IN THE EXPERIMENTAL DATASETS

	iTrust		WDS	
	Dep.	Non-D.	Dep.	Non-D.
Trace granularity	Use cases → Java methods		Feature requests → Java classes	
# of req.s	13	22	11	171
Average # of true links	8.69	8.77	9.45	10.99
Range of true links	2–39	3–44	3–19	2–123

TABLE II
ITRUST DEPENDABILITY REQUIREMENTS

REQs	Type
UC1 create and disable patients	AC, UUI
UC2 create, disable, and edit personnel	AC, UUI
UC3 authenticate users	PA, AL, INT
UC5 log transaction	TS
UC8 view access log	TS, AUD
UC9 view records	AUD
UC18 maintain a hospital listing	UUI
UC21 view emergency electronic health record	AUD
UC23 view comprehensive patient report	AUD
UC25 view physician satisfaction survey results	AUD
UC28 view patients	AUD
UC32 proactively confirm prescription-renewal needs	AUD
UC38 maintain drug interaction	UUI

TABLE III
WDS DEPENDABILITY REQUIREMENTS

REQs	Type
Add database connection	Integrity, safety
Input/edit jobs	Availability
Job validation	Safety
Job import	Safety
Job export	Safety
Input/edit training	Availability
Training validation	Safety
Godfather role add edit rights	Integrity
Search flow	Reliability, maintainability
Edit preferences	Safety
WIA service	Availability

not have the ability to enter/edit/view the patient's security question/password." We, therefore, mark Use Case 1 as an AC (access control) requirement. As shown in Tables II and III, each dependability requirement is classified into one or more categories. Figs. 2 and 3 show the taxonomies where the dependability categories are fully defined.

We compare our term-based RF mechanism with three other requirements tracing methods: the TF-IDF vector space model [6] serving as the baseline, the standard Rocchio algorithm [14], and the adaptive Rocchio variant [8]. We evaluate the tracing methods along three dimensions: effectiveness, browsability, and specificity. The standard metrics of IR-based tracing effectiveness are: recall, precision, and F_2, as defined in (1) and (2). Browsability of the resulting ranked list of the traceability links complements the effectiveness measures

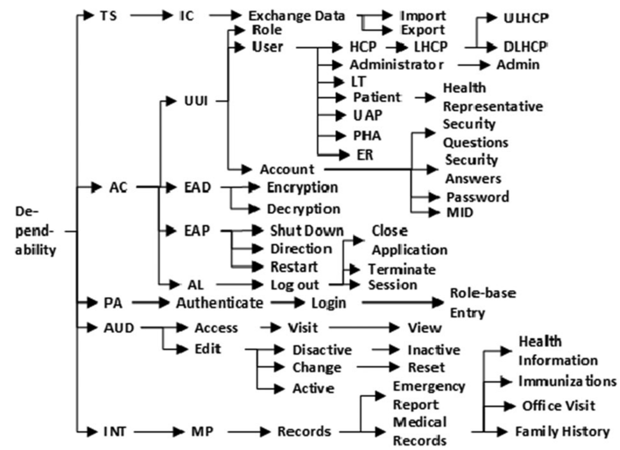


Fig. 2. iTrust dependability taxonomy: TS (transmission security), AC, INT (integrity), PA (person or entity authentication), AUD (audit controls), IC (integrity control), AL (automatic logoff), UUI (unique user identification), EAP (emergency access procedure), EAD (encryption and decryption), MP (mechanism to authenticate electronic protected health information), ER (emergency responder), HCP (health care professional), UAP (unlicensed authorized personnel), PHA (public health agent), LT (lab technician), LHCP (licensed HCP), DLHCP (designated LHCP), UL-HCP (unlicensed LHCP).

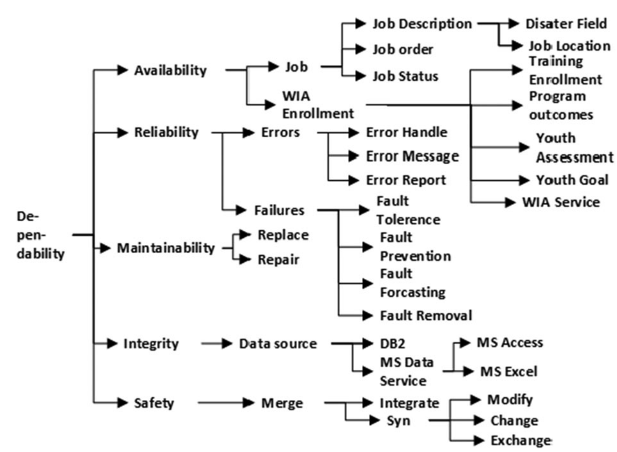


Fig. 3. WDS dependability taxonomy: WIA (workforce investment act).

because recall, precision, and F_2 are all set-based metrics. Following [6], we adopt two browsability metrics: mean average precision (MAP) and Lag. We next describe a new metric to quantify the specificity of RF mechanism in the context of dependability requirements tracing.

The central idea of our specificity metric is to assess the extent to which a requirement expresses dependability needs. To that end, we first manually build a dependability taxonomy for each domain by following the grounded-theory approach presented in [19]: Fig. 2 for iTrust and Fig. 3 for WDS. From left to right, the degree of specificity increases. For example, "close application" and "terminate" are more specific than "log out" in Fig. 2. Note that each taxonomy is constructed independent of tracing methods, especially the RF mechanism. We then leverage the taxonomy to formulate the ideal dependability representation of a particular requirement, which we denote as R_{dep}. Finally, we compare R_{dep} with the requirement's representation resulted from RF in order to calculate specificity.

TABLE IV
RF PERFORMANCES IN TRACING ITRUST'S DEPENDABILITY REQUIREMENTS (DEP. REQ.S) AND NON-DEPENDABILITY REQUIREMENTS (NON-D. REQ.S)

(a) DESCRIPTIVE STATISTICS

		TF-IDF		standard Rocchio		adaptive Rocchio		term-based RF	
		Dep. Req.s	Non-D. Req.s	Dep. Req.s	Non-D. Req.s	Dep. Req.s	Non-D. Req.s	Dep. Req.s	Non-D. Req.s
Effec-tive-ness	*Recall*	0.79	0.76	0.83	0.86	0.84	0.93	0.94	0.90
	Precision	0.07	0.09	0.09	0.08	0.10	0.14	0.16	0.12
	F_2	0.25	0.30	0.31	0.30	0.34	0.44	0.48	0.40
Brows-ability	*MAP*	0.29	0.28	0.35	0.30	0.38	0.39	0.42	0.38
	Lag	215.89	204.37	156.28	147.26	98.61	83.71	76.25	86.48
Specificity		27.65	–	26.43	–	23.33	–	10.34	–

(b) INFERENTIAL STATISTICS

	Effectiveness			Browsability		Specificity
	Recall	Precision	F_2	MAP	Lag	
SR vs TF-IDF	33.24	16.3 ($\hat{A}_{12}$=0.60)	50.39	30.97	58.86	127.5
AR vs TF-IDF	10.3 ($\hat{A}_{12}$=0.76)	9.21 ($\hat{A}_{12}$=0.84)	24.97	1.95 ($\hat{A}_{12}$=0.86)	0.48 ($\hat{A}_{12}$=0.71)	9.48 ($\hat{A}_{12}$=0.72)
AR vs SR	146.3	62.73	63.31	100.3	19.6	224.7
TB vs TF-IDF	6.34 ($\hat{A}_{12}$=0.80)	4.73 ($\hat{A}_{12}$=0.88)	6.32 ($\hat{A}_{12}$=0.84)	0.23 ($\hat{A}_{12}$=0.88)	0.14 ($\hat{A}_{12}$=0.78)	2.61×10^{-5} ($\hat{A}_{12}$=0.88)
TB vs SR	11.4 ($\hat{A}_{12}$=0.66)	6.38 ($\hat{A}_{12}$=0.84)	17.98	12.98	2.68 ($\hat{A}_{12}$=0.65)	0.056 ($\hat{A}_{12}$=0.84)
TB vs AR	12.9 ($\hat{A}_{12}$=0.58)	7.32 ($\hat{A}_{12}$=0.89)	19.36	20.35	392.2	0.061 ($\hat{A}_{12}$=0.86)

Due to space constraint, only dependability requirements' results are listed. The reported p value is in the 10^{-3} scale. SR: standard Rocchio, AR: adaptive Rocchio, TB: term-based RF.

Because RF essentially adjusts the weight of a requirement's terms and/or adds new terms to the query requirement, our R_{dep} construction has two phases: manipulating existing term's weight and appending the requirement with new query terms. We illustrate the two phases with iTrust's Use Case 9 ("view records"). From Table II, this requirement is an AUD (audit controls) requirement. Therefore, the subtree of AUD in Fig. 2 plays a key role in defining this requirement's R_{dep}.

In the first stage, all the terms of Use Case 9 are checked, and if the term belongs to the AUD subtree, then we use the log-scale of the term's depth [20] to adjust the term's weight. The terms "visit" and "view" appear in both the original description of Use Case 9 and the AUD subtree. They are subject to weight increase. The TF-IDF value of "visit" and "view" are 0.47 and 0.79, respectively. The log-scale term depth in the AUD subtree is log(4) $=$ 0.60 for "visit" and log(5) $=$ 0.69 for "view." Thus, after the first stage, the adjusted weight of "visit" and "view" is 1.07 and 1.48, respectively. The second phase takes the terms belonging to the AUD subtree and then adds them with their log-scale depth weights to the R_{dep} of Use Case 9. For instance, "edit" is appended with the weight of log(3) $=$ 0.48. Note that the weight of "visit" and "view" is adjusted again in stage two, making 1.67 and 2.17 their final weights in R_{dep}.

The resulting R_{dep} is a new vector (w_{t_1}, w_{t_2}, ..., w_{t_m}) where m is the number of total terms in the domain vocabulary. According to the dependability type of the requirement and the taxonomy related to the type, we consider R_{dep} best reflects the dependability concerns of the requirement, and thus, refer it to the ideal query requirement representation. Using R_{dep}, we define *specificity* as follows:

$$\text{Specificity} = \sqrt{\sum_{i}^{m} (w_{t_i} - w'_{t_i})^2} \tag{4}$$

where w_{t_i} is the term weight in R_{dep} and w'_{t_i} is the term weight in the dependability requirement adjusted by a given RF mechanism. Thus, specificity computes the distance between a requirement's representation undergone RF and its ideal representation reflecting the particular dependability concerns. The less the specificity value according to (4), the better the RF algorithm in transforming the dependability requirement to its ideal representation.

B. Results

We analyze the experimental results based on our three evaluation goals: effectiveness, browsability, and specificity. Our analyses are divided into dependability requirements (i.e., those in Tables II and III) and non-dependability requirements, following the dependability requirements classification described in Section IV-A. We preprocessed the software artifacts in a uniform way. In particular, an expanded indexer that handles both natural-language requirements and Java source code was deployed [9]. For RF, we instantiated the parameters as $\alpha = 1.0$, $\beta = 0.75$, and $\gamma = 0.25$ because these values exhibit consistency and robustness in both traditional IR [15] and automated traceability [6], [8].

Arguably, effectiveness is one of the most important criteria used to evaluate IR-based tracing methods. Metrics like recall, precision, and F_2 provide measures toward the tracing algorithm's accuracy as well as the automated tool's believability [6]. Table IV-a presents the descriptive statistical results on the iTrust dataset when the 70% threshold is applied to the tracing methods (i.e., only evaluating the top 70% of retrieved candidate links). To compare the performance of the four different tracing methods, a pairwise Bonferroni–Holm correction is conducted and the effect sizes are computed using $\hat{A}_{12}$ non-parametric statistics [21]. The inferential results are reported in Table IV-b, where statistically insignificant p-values are given without $\hat{A}_{12}$ values. Following [21], we use the intervals defined by 0.56, 0.64, and 0.71 to distinguish small, medium, and large effect sizes.

Table IV-b shows that, compared with the baseline TF-IDF method, our term-based RF algorithm achieves significantly better performances in all the three areas of effectiveness,

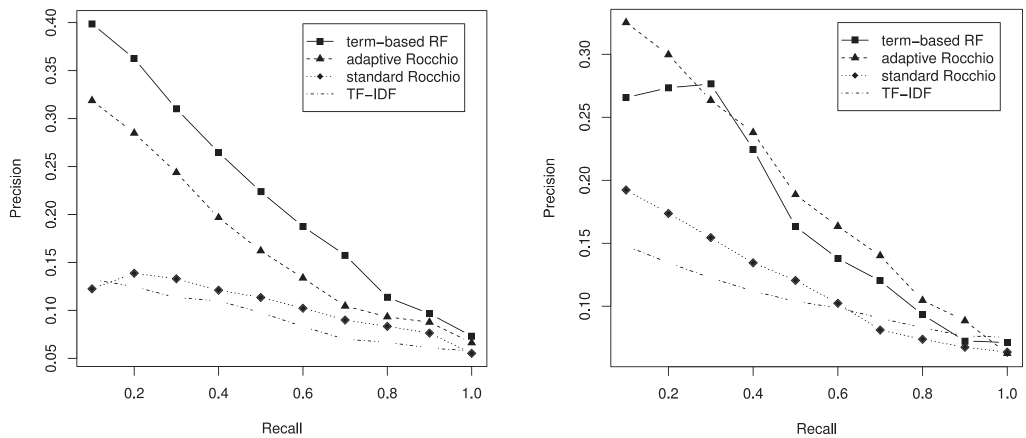


Fig. 4. Recall-precision analysis of iTrust: dependability requirements tracing (left); non-dependability requirements tracing (right).

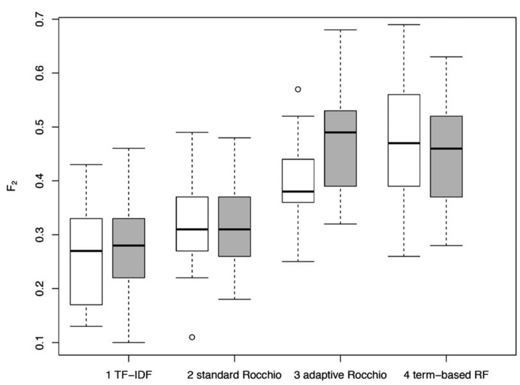


Fig. 5. Tracing effectiveness on WDS.

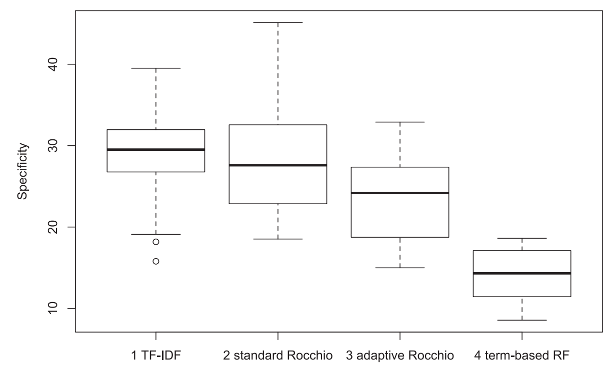


Fig. 6. Specificity of WDS dependability requirements.

browsability, and specificity. Furthermore, the effect sizes are large. Our method also outperforms the standard and adaptive Rocchio in a significant manner. This trend can be readily visualized in Fig. 4 where 10 cutoff points (0.1, 0.2, ..., 1.0) are applied on recall.

When tracing non-dependability requirements, our term-based RF algorithm has mixed performances when compared with the adaptive Rocchio method. As shown in the right of Fig. 4, term-based RF has a plateau at the low recall values. We conjecture the main reason is because, for non-dependability requirements, their top five most important terms are often too general. For example, iTrust's UC 17 ("proactive determine needed patient") returns "patient," "month," "week," "immunization," and "alphabetical." Fine-tuning the weights of these terms, as opposed to performing link-based RF like standard or adaptive Rocchio, leads to more false links to be ranked higher in the tracing results, e.g., the links containing "patient name" and "alphabetical sort." In this sense, dependability requirements are more suited for the term-based RF treatment (cf. left of Fig. 4). The tracing effectiveness of WDS exhibits the same trend as iTrust. Due to the space constraint, only the F_2 statistics are summarized in Fig. 5.

Similarly to effectiveness, the browsability measures show the superior performance of our term-based RF method over the other three methods. For specificity, only the dependability requirements are compared among the four tracing methods. Recall that specificity captures the distance between a dependability requirement's ideal representation (R_{dep}) and its representation adjusted by RF. For TF-IDF, no RF is applied, and therefore, specificity is the lowest. Surprisingly, standard and adaptive Rocchio do not improve specificity significantly, as shown by the results of Table IV and Fig. 6. In contrast, specificity is enhanced by term-based RF, indicating that the trace query resulted from our algorithm is the closest to the dependability concerns that the requirement intends to express.

C. Threats to Validity

We mitigate the threats to *construct validity* [22] by considering three performance facets: effectiveness, browsability, and specificity. For the first two, we adopt standard IR metrics [10]. For specificity, one limitation is our manual construction of the domain-specific dependability taxonomy. While ontology engineering has dramatically advanced in the past several decades, fully automated ways to build the knowledge base for dependability hardly exist. Even though researchers may be eager to advocate their exciting ontology building techniques, we argue that certain level of manual intervention, like quality control or consistency management, is unavoidable. In this sense, the taxonomies of Figs. 2 and 3 should be regarded only as starting points and are subject to subsequent refinement and maintenance. While this limitation should have little effect on the comparisons, caution must be taken in interpreting the absolute values of specificity.

We believe the main strength of our experimental design is its high *internal validity* [22]: soundness of the relationship between independent and dependent variables. Because all the factors potentially affecting the responses (effectiveness, browsability, and specificity) are under our direct control, any significant difference must be caused by the different requirements tracing methods employed.

The results of our analysis may not generalize to other dependability requirements tracing datasets—a threat to the *external validity* [22]. Our chosen systems cover both safety-critical and mission-critical applications. However, these are not necessarily representative of all dependable industrial systems and, in particular, embedded software products are likely to exhibit different characteristics.

V. Discussion

An analytical comparison pinpointing the theoretical improvements of RF mechanisms over the baseline TF-IDF method is as follows: standard Rocchio [14] adjusts the weights for the candidate traceability links, adaptive Rocchio [8] adjusts those for only a subset of the links, and our approach performs weight adjustment for only the selected terms within each link.

One of the key results is that our algorithm outperforms the standard and adaptive Rocchio in tracing dependability requirements. An explanation is that the two Rocchio methods instrument the RF at the link level, that is, if a candidate traceability link is marked as a positive (or negative) feedback, then *all* the terms of that link will be treated as positive (or negative) to modify the query requirement.

In contrast, our method works at the term level, which allows for different treatments of different terms. In our algorithm presented in Fig. 1, the requirements term is checked against its usage in the code base. When the gram containing the term is used in a regular and repetitive way in code, we then analyze the term's context in the requirement and use this information to determine the RF type: positive, negative, or uncertain (unchanged). The main benefit of the term-based RF algorithm, in our opinion, is the finer-grained control over the term weighting, especially for the terms that appear in the same requirement.

It turns out that dependability requirements are commonly expressed with specific terms, and these terms are not only domain specific but also concern specific [23] and task specific [24]. For these reasons, we constructed two taxonomies for the systems that we studied, and further used the taxonomies to define a new metric to measure specificity. The experimental results show that our term-based RF method was able to transform the original query requirement into a form that is the closest to the requirement's dependability concerns. In this sense, the superior performances of our algorithm in effectiveness and specificity are not independent, but correlated. Finally, it is worth mentioning that our term-based RF method can potentially improve the tracing of other nonfunctional requirements (such as portability and interoperability) than dependability, as well as those concerns related to dependability (such as safety and reliability). Testing the effect of term-based RF on other concerns requires further experiments, and if specificity is to be assessed using our method, new taxonomies or other knowledge representations of concern-dependent terms.

The implications of our work are two-fold. For researchers working in requirements traceability, our algorithm represents a significant departure from applying traditional methods in IR. In fact, one of the latest developments, namely the adaptive Rocchio method [8], critically revisits the underlying assumptions of RF and modifies how Rocchio should be operated in requirements tracing. Similarly, the method proposed in this paper, to the best of our knowledge, is the first algorithm that automates RF at the term level. Such a novelty combines the naturalness of software [17] and the specificity of dependability requirements. More importantly, we anticipate our work to illuminate the practitioners with further automation to reduce the manual effort in engineering dependable industrial systems. As is commonly believed, one cannot patch security—for the same reason, dependability—late in the development life cycle. The most cost-effective stage of building dependability is, therefore, in requirements engineering. Our work illustrates that engineering dependability requirements is not only about elicitation. One also needs V&V and our algorithm represents a new way to achieve both effectiveness and automation. We, therefore, encourage researchers and practitioners to go beyond traditional domains like IR so that transformative innovations can be made to best fit the critical tasks of requirements analysts, software developers, system assurers, and other industrial engineers.

VI. Conclusion

In industrial informatics, software has become an increasingly important enabler to deliver dependability in various applications ranging from power infrastructures to medical devices. The dependability concerns shall be rigorously engineered in the requirements phase to better inform the entire life cycle of the system development. In this paper, we have proposed a novel term-based RF mechanism that automates some activities related to the V&V of dependability requirements. Experimental evaluation of two systems shows that our method outperforms the contemporary link-based RF solutions (namely standard and adaptive Rocchio).

Our future work includes assessing the usability of our method, conducting more empirical evaluations with different types of industrial systems, covering other types of dependability concerns like fault tolerance and autonomic healing, expanding the tracing to other nonfunctional requirements such as maintainability and reliability, and incorporating ontology engineering to refine and reuse the knowledge representation and reasoning of dependability.

References

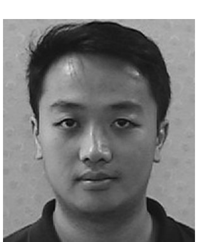
[1] M. Pajic, R. Mangharam, O. Sokolsky, D. Arney, J. M. Goldman, and I. Lee, "Model-driven safety analysis of closed-loop medical systems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 3–16, Feb. 2014.

[2] R. Muradore and D. Quaglia, "Energy-efficient intrusion detection and mitigation for networked control systems security," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 830–840, Jun. 2015.

[3] F. Luo *et al.*, "Advanced pattern discovery-based fuzzy classification method for power system dynamic security assessment," *IEEE Trans. Ind. Informat.*, vol. 11, no. 2, pp. 416–426, Apr. 2015.

[4] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Depend. Sec. Comput.*, vol. 1, no. 1, pp. 11–33, Jan./Mar. 2004.

[5] N. Hussein, W. Wang, J. L. Nedelec, X. Wei, and N. Niu, “Unified profiling of attackers via domain modeling,” in *Proc. Int. Workshop Requirements Eng. Investigating Countering Crime*, Beijing, China, Sep. 2016, pp. 98–101.
[6] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, “Advancing candidate link generation for requirements tracing: The study of methods,” *IEEE Trans. Softw. Eng.*, vol. 32, no. 1, pp. 4–19, Jan. 2006.
[7] N. Niu, A. Mahmoud, Z. Chen, and G. Bradshaw, “Departures from optimality: Understanding human analyst’s information foraging in assisted requirements tracing,” in *Proc. Int. Conf. Softw. Eng.*, San Francisco, CA, USA, May 2013, pp. 572–581.
[8] A. Panichella, A. De Lucia, and A. Zaidman, “Adaptive user feedback for IR-based traceability recovery,” in *Proc. Int. Symp. Softw. Syst. Traceability*, Florence, Italy, May 2015, pp. 15–21.
[9] N. Niu and S. Easterbrook, “Extracting and modeling product line functional requirements,” in *Proc. Int. Requirements Eng. Conf.*, Barcelona, Spain, Sep. 2008, pp. 155–164.
[10] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge Univ. Press, 2008.
[11] S. A. Asghari, H. Taheri, H. Pedram, and O. Kaynak, “Software-based control flow checking against transient faults in industrial environments,” *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 481–490, Feb. 2014.
[12] N. Niu and S. Easterbrook, “Analysis of early aspects in requirements goal models: A concept-driven approach,” *Trans. Aspect-Oriented Softw. Develop.*, vol. 3, pp. 40–72, 2007.
[13] N. Niu, Y. Yu, B. González-Baixauli, N. Ernst, J. Leite, and J. Mylopoulos, “Aspects across software life cycle: A goal-driven approach,” *Trans. Aspect-Oriented Softw. Develop.*, vol. 6, pp. 83–110, 2009.
[14] J. J. Rocchio, “Relevance feedback in information retrieval,” in *The SMART Retrieval System: Experiments in Automatic Document Processing*, G. Salton, Ed. Englewood Cliffs, NJ, USA: Prentice Hall, 1971.
[15] I. Ruthven and M. Lalmas, “A survey on the use of relevance feedback for information access systems,” *Knowl. Eng. Rev.*, vol. 18, no. 2, pp. 95–145, Jun. 2003.
[16] Y. Shin and J. Cleland-Huang, “A comparative evaluation of two user feedback techniques for requirements trace retrieval,” in *Proc. ACM Symp. Appl. Comput.*, Trento, Italy, Mar. 2012, pp. 1069–1074.
[17] A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, “On the naturalness of software,” in *Proc. Int. Conf. Softw. Eng.*, Zurich, Switzerland, Jun. 2012, pp. 837–847.
[18] W. Wang, N. Niu, H. Liu, and Y. Wu, “Tagging in assisted tracing,” in *Proc. Int. Symp. Softw. Syst. Traceability*, Florence, Italy, May 2015, pp. 8–14.
[19] T. Scott, K. Kuksenok, D. Perry, M. Brooks, O. Anicello, and C. Aragon, “Adapting grounded theory to construct a taxonomy of affect in collaborative online chat,” in *Proc. Int. Conf. Design Commun.*, Seattle, WA, USA, Aug. 2012, pp. 197–204.
[20] P.-M. Ryu and K.-S. Choi, “Taxonomy learning using term specificity and similarity,” in *Proc. Workshop Ontology Learn. Population*, Sydney, Australia, Jul. 2006, pp. 41–48.
[21] A. Vargha and H. Delaney, “A critical and improvement of the CL common language effect size of mcgraw and wong,” *J. Educational Behavioral Statist.*, vol. 25, no. 2, pp. 101–132, 2000.
[22] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. New York, NY, USA: Springer, 2012.
[23] N. Niu, L. D. Xu, and Z. Bi, “Enterprise information systems architecture—analysis and evaluation,” *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2147–2154, Nov. 2013.
[24] N. Niu, W. Wang, and A. Gupta, “Gray links in the use of requirements traceability,” in *Proc. ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Seattle, WA, USA, Nov. 2016, pp. 384–395.

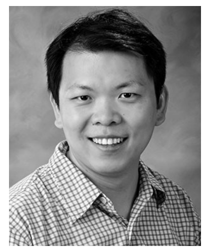


Wentao Wang (S’15) received the B.Sc. degree in computer science from Shanghai Maritime University, Shanghai, China, in 2007, the M.Eng. degree in software engineering from Beijing Institute of Technology, Beijing, China, in 2010, and is currently working toward the Ph.D. degree in the Department of Electrical Engineering and Computing Systems, University of Cincinnati, Cincinnati, OH, USA.

His research interests include software requirements engineering, information seeking in software engineering, and information retrieval.

Arushi Gupta is currently working toward the Bachelor’s degree in Computer Engineering the Department of Electrical Engineering and Computing Systems, University of Cincinnati, Cincinnati, OH, USA.

She has been working in the Software Engineering Research Lab, University of Cincinnati, for two years. Her research interest focuses on software and systems traceability.



Nan Niu (M’08–SM’13) received the B.Eng. degree from the Beijing Institute of Technology, Beijing, China, in 1999, the M.Sc. degree from the University of Alberta, AB, Canada, in 2004, and the Ph.D. degree from the University of Toronto, Toronto, ON, Canada, in 2009, all in computer science.

He is currently an Assistant Professor in the Department of Electrical Engineering and Computing Systems, University of Cincinnati, Cincinnati, OH, USA. His research interests include software requirements engineering, information seeking in software engineering, and human-centered computing.

Dr. Niu received the US National Science Foundation Faculty Early Career Development Award.

Li Da Xu (M’86–SM’11–F’16) received the M.S. degree in information science and engineering from the University of Science and Technology of China, Hefei, China, in 1981, and the Ph.D. degree in systems science and engineering from Portland State University, Portland, OR, USA, in 1986.

He serves as the Founding Chair of IFIP TC8 WG8.9 and the Founding Chair of the IEEE SMC Society Technical Committee on Enterprise Information Systems.

Jing-Ru C. Cheng received the Ph.D. degree in computer science from Penn State University, State College, PA, USA, in 2002.

Since 2002, she has been a Computer Scientist with the US Army Engineer Research and Development Center, Vicksburg, MS, USA, . Her research interests include parallel algorithm development, software tool development for scientific computing, and multiscale multiphysics code development.

Zhendong Niu received the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 1995.

He is a Professor and Deputy Dean in the School of Computer Science and Technology, Beijing Institute of Technology. His research interests include informational retrieval, software architecture, digital libraries, web-based learning techniques, and more.

Dr. Niu received the IBM Faculty Innovation Award in 2005 and was also awarded the New Century Excellent Talents in the University of MOE of China in 2006.