Predicting the Vector Impact of Change - An Industrial Case Study at Brightsquid

Shaikh Jeeshan Kabeer, Maleknaz Nayebi, Guenther Ruhe
SEDS Laboratory, University of Calgary
Calgary, Canada
Email: {shaikhjeeshan.kabeer,mnayebi,ruhe}@ucalgary.ca

Chris Carlson, Francis Chew
Brightsquid
Calgary, Canada
Email: {chris.carlson,francis}@brightsquid.com

***Abstract*—*Background*: Understanding and controlling the impact of change decides about the success or failure of evolving products. The problem magnifies for start-ups operating with limited resources. Their usual focus is on Minimum Viable Product (MVP's) providing specialized functionality, thus have little expense available for handling changes. *Aims:* Change Impact Analysis (CIA) refers to the identification of source code files impacted when implementing a change request. We extend this question to predict not only affected files, but also the effort needed for implementing the change, and the duration necessary for that. *Method:* This study evaluates the performance of three textual similarity techniques for CIA based on Bag of words in combination with either topic modeling or file coupling. *Results:* The approaches are applied on data from two industrial projects. The data comes as part of an industrial collaboration project with Brightsquid, a Canadian start-up company specializing in secure communication solutions. Performance analysis shows that combining textual similarity with file coupling improves impact prediction, resulting in Recall of 67%. Effort and duration can be predicted with 84% and 72% accuracy using textual similarity only. *Conclusions:* The relative effort invested into CIA for predicting impacted files can be reduced by extending its applicability to multiple dimensions which include impacted files, effort, and duration.**

***Keywords*-Change Impact Analysis; Case Study; Software Repository; Topic Modeling; Bag of Words; Effort estimation**

I. Introduction

Fulfilling software requirements is of paramount importance to software success. Requirements reflect and capture the intended goals and objectives of the products. However, one release does not satisfy all the requirements. More and more modifications are requested by stakeholders, which appear in the form of Change Requests (CRs) during the system development and maintenance process [19]. CRs appears primarily due to changes in demands, stakeholders, market, tools and technology [14], [22]. These requests need to be handled efficiently to minimize software cost and enhance quality [27]. Better understanding and management of change requests in start-up companies with market pressure is of critical importance.

Brightsquid's Secure Communication Corp. is a global provider of HIPAA-compliant communication solutions providing messaging, email, and large file transfer for medical and dental professionals since 2009. Change request management is of core importance in this highly adaptive process. Often, change requests translates to development activities on separate repositories.

For a start-up company like Brightsquid, which is operating with limited resources, creating Minimum Viable Products (MVP) is a promising pathway to enter the market in shorter time [15]. MVPs are accelerated products designed to hit the market quickly, elicit customer feedback and also announce the presence in the market [24]. In this fast-paced development environment, one of the vital components is scoping changes and estimating effort and time needed for that change.

Change Impact Analysis (CIA) is concerned with identifying source code files which will be impacted by implementing a change request. Textual similarity and file coupling have been used in CIA, some of the most popular ones are [9], [10], [16], [18], [26], [28]. Similarly, topic modeling is a widely used technique in software change analysis [12]. In this study, we used analogy based reasoning for estimating the scope of change, effort and time needed to apply the change. We use textual similarity and topic modeling to define analogy between change requests. For better predicting the scope of change (CIA), we used coupling between files in addition to the textual similarity. While there are many applications of CIA on open source data, only a few studied such as [1], [8] are known which report the results from an industrial context.

The objective of this paper is (i) to analyze the performance of three textual similarity based techniques for CIA in an industrial context and (ii) to extend the usage of the similarity models created for (i) to also predict the efforts and duration of CRs. In total, we consider CIA for predicting impact files (called CIA(F)), the effort of implementing a CR (called CIA(E)), and duration of implementing a CR (called CIA(D)).

We analyzed the below research questions:

RQ1: Among the three approaches (i) Bag of words (ii) Bag of words and topic modeling and (iii) Bag of words and file coupling, which one works best better for predicting impacted files by a change request? How the techniques perform in dependence of the number of similar objects considered and in dependence of the type of change request?

RQ2: How well can textual similarity as defined in RQ1 also be used for predicting effort and duration of implementing a change request?

Section II describes the industrial collaboration and mo-

tivation for this study. In Section III we describe the case study design. Section IV and Section V present the key results and comparison of results to approaches defined in literature respectively. Section VI and Section VII discuss the lessons learned and threats to validity.

II. Context and Motivation

We conducted this study at the software development unit of Brightsquid, a growing start-up in the field of health informatics. Brightsquid provides solutions for clinical communications between doctors, patients, and clinic support staff. Their core product is `Secure-Mail` which is a secure communication and collaboration platform. This platform allows convenient patient care management, sharing results, and allocate expertise very quickly with thousands of users.

Brightsquid follows agile (Scrum) methodology for software development. Brightsquid uses `GitHub` for version control and `Jira` for issue management. Their products are implemented, refined and maintained incrementally through multiple sprints, each of duration two weeks. Software development at Brightsquid happens in a complex ecosystem. Their core product is a secure messaging platform which is offered as the web, desktop and mobile application services. Different software functionality is implemented across thousands of files, resulting in a complicated system of interacting entities.

For scoping of the collaboration, we performed a survey following the structure introduced by Begel and Zimmermann [6]. The participants were asked to identify the most critical questions from their perspective, which they would like answered. We received the response from seven Brightsquid staff (100% response rate). A total of 21 questions were stated to be most important for Brightsquid. Using open card sorting [6], the questions were categorized into "Defects and testing", "Productivity", "Code usage and solubility", and "Processes and Standards". Each question could belong to more than one of these four categories.

About one fourth of these questions (23.8%) were related to change request analysis and the subsequent actions:

- What is the general cost of change on this software in comparison with accepted standard?
- Which generates more test cases, a new feature or a change request?
- Which leads to more defects: new features or a change request?
- What's the impact of change requests and tickets on the "performance" of the teams for the company?
- How can we find the most important areas of our code base?

We took a first step toward answering these questions by applying analogy based reasoning. In this paper, we focused on studying the extent to which textual similarity can be used to infer analogy between change requests. We evaluated the impact of textual similarity on the modified files (change impact analysis), on the effort (effort estimation by analogy) and timing (scheduling by analogy).

Figure 1 shows the change management process at Brightsquid. In each release, a set of change requests is selected by Product Manager. The project manager assigns each change to a developer and estimates the time needed to apply the change. Depending on nature, each request might require changes to one or multiple files. The files requiring change might be very different in terms of operation, location, and functionality. One of the key tasks of the Developer is to analyze the changes requested and identify which files would require modifications and the corresponding effort and duration to implement the change. The identification of files, effort, and duration is mostly subject to expert knowledge and experience of solving similar problems. It requires careful navigation and expertise of the complex interaction of the code entities. Also, due to the size of the system, it is inefficient, even for experienced developers to locate the scope of these changes manually. The dashed box labeled Change Impact Analysis (CIA) in Figure 1, is where a semi-automated solution would be most helpful towards change identification and implementation.

III. Case Study Design

The overall study process of this paper is outlined in Figure 2. In this section, we describe the major steps of the process.

A. Data Retrieval and Pre-processing

We created a corpus using the summary and description of the change requests that we mined from `Jira`. The summary is a title given to each CR while the description describes the request in full detail. We analyzed two projects of Brightsquid called `Mail` and `Dental`. Table I summarizes the data extracted from the repositories. For this paper, projects `Mail` and `Dental` were retrieved and analyzed from `Jira`. `Mail` is a more recent project with active tickets, while `Dental` is

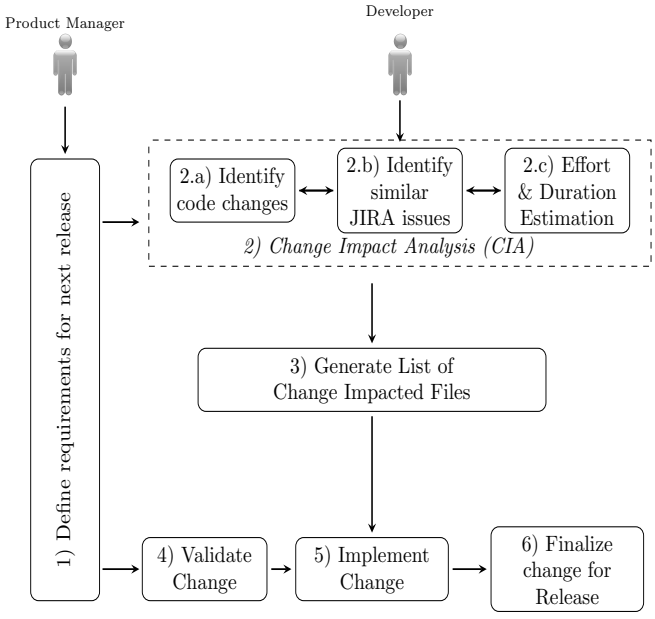

Fig. 1. Change Management Process at Brightsquid

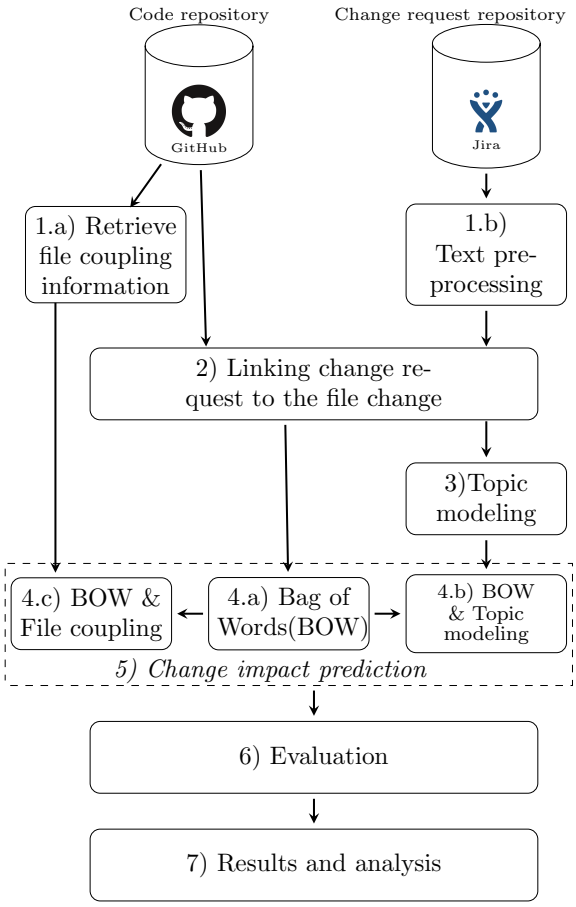

Fig. 2. Process flowchart of Vector CIA

an older project in which no new tickets are opened anymore. The duration of analysis is between $July, 2015$ to $May, 2017$.

The `Mail` project has 1716 change requests between May 2016-2017 and 685 of them are traceable in `GitHub` repository. The links between CR and files changed(in `GitHub`) are identified by analyzing commit messages, which contains the CR ID used for establishing traceability. Traceability was established similar to traditional approaches such as [25]. This is step 2 of the process outlined (Figure 2). These 656 change requests impacted 1,227 files overall. `Dental` project has 648 change requests between July 2015-2016, with 370 of them being traceable to files changed. These 370 change requests impacted 1,458 files. For the time duration of each specified project, *#CR* under Effort Prediction represent the number of change requests for which effort estimations were available in `Jira`. Similarly, *#CR* under Duration Prediction represent the number of CR for which the duration information was available.

TABLE I
TIME FRAME AND NUMBER OF JIRA ISSUES FOR FILE, EFFORT, AND DURATION PREDICTIONS

Data Source	Duration	File	Effort	Duration
		# CR	# CR	# CR
Mail	May 2016 - May 2017	685	542	1116
Dental	July 2015- July 2016	370	200	528

We took the following steps for pre-processing, the first step was *Basic text cleaning*, where we removed the special characters, hyperlinks, and email addresses. Next, *Removing Stop Words* operation was performed, where we removed commonly occurring words. Finally, we applied Porter's stemming algorithm [23] to put the words in their root dictionary format.

B. Similarity Measure

Cosine similarity is one of the most popular techniques for measuring the textual similarity between two documents [3]. It has been applied to a wide spectrum of information retrieval applications including CIA. We calculated the similarity between two change requests using Cosine Similarity of their textual content. TF-IDF (term frequency - inverse term frequency) is a statistical measure used to evaluate the importance of words in a collection of documents. It consists of two components, *Term Frequency (TF)*, which is a count of the number of times a word appears in a document, normalized by the total number of words in that document. The second component is the *Inverse Document Frequency (IDF)* which is the logarithm of the number of documents in the corpus divided by the number of documents where the particular term appears. We used TF-IDF to represent CR in the vector space.

Representing each CR as a vector in vector space, The similarity S between CRs is calculated as follows:

$$S(q,d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|} \quad (1)$$

where q and d are the query and the existing CR, respectively. $\vec{V}(q) \cdot \vec{V}(d)$ represents the dot product of the weighted vectors with $|\vec{V}(q)||\vec{V}(d)|$ being their Euclidean norms.

C. Impact Prediction Techniques

This paper analyzes the performance of three techniques (step 4 of Figure 2), used in different configurations for the different types of problems. To keep the paper self-contained, we briefly describe the techniques.

1) Bag of Words (BOW): The base model is the Bag of Words (BOW), a data representation technique, particularly used in natural language and information retrieval applications [21]. BOW represents textual data as multi-set of constituent words, without their ordering or other attributes. The main idea of BOW is to construct a knowledge base consisting of existing changes.

For this model, each CR is represented as a BOW entity consisting of corresponding CR Summary and Description. From the data retrieved from repositories the corpus is created, which reflects the knowledge base against which new change requests will be evaluated. Using this knowledge base, file impacts for new change requests are predicted following steps of Algorithm 1. The algorithm takes as input all change requests in the knowledge base and the query CR (CR_N and

CR_q respectively). The query CR represents the new CR, for which file impact needs to be predicted. The similarity is calculated between each query and existing CRs. If threshold T is satisfied, the corresponding CR is marked as similar. For all available similar CRs SCR found from existing data, their past *file changes, effort and duration* (sr_F, sr_e, sr_d respectively) are added to final results. For a (new) query change request, output is a list of impacted files, effort and duration called I_F, I_E, I_D respectively.

Algorithm 1 Bag of Words similarity for File, Duration and Effort prediction

Input: CR_q, CR_N in
Output: I_F, I_E, I_D out
1: **for** $i = 1$ to N **do**
2: Calculate $S(CR_q, CR_i)$ {using equation 1}
3: **if** $(S > T_B)$ **then**
4: Add CR_i to list of similar CRs SCR
5: **end if**
6: **for** each element $sr \in SCR$ **do**
7: Add all $f \in sr_F$ to I_F
8: Add sr_e to I_E
9: Add sr_d to I_D
10: **end for**
11: **end for**
12: **return** I_F, I_E, I_D

Algorithm 2 BOW & Topic Modeling similarity for File, Duration and Effort Prediction

Input: CR_q, CR_N, CR_{qT}
Output: I_F, I_E, I_D
1: **for** $i = 1$ to K **do**
2: Calculate $S(CR_{qT}, CRT_{iT})$ {using equation 1}
3: **if** $(S > T_M)$ **then**
4: For each cr with similar topic terms, add to SL
5: **end if**
6: **for** each $sl \in SL$ **do**
7: Add all $f \in sl_f$ to I_{TM}
8: **end for**
9: **end for**
10: **for** $i = 1$ to N **do**
11: Calculate $S(CR_q, CR_i)$ {using equation 1}
12: **if** $(S > T_B)$ **then**
13: Add CR_i to list of similar CRs SCR
14: **end if**
15: **for** each $sr \in SCR$ **do**
16: Add all $f \in sr_f$ to I_{BOW}
17: Add sr_e to I_E
18: Add sr_d to I_D
19: **end for**
20: **end for**
21: $I_F = I_{BOW} \cap I_{TM}$
22: **return** I_F, I_E, I_D

2) BOW & Topic Modeling (BOW & TM): Latent Dirichlet Allocation (LDA) topic modelling [7] is applied on all available change request CR_N (pre-processed Summary and Description text) to generate Z_K topics, where membership of topic $z_k \in Z_K$ for change request $cr_i \in CR_N$ is $\theta(cr_i, z_k)$, $\forall i, k : 0 \leq \theta(cr_i, z_k) \leq 1$ and $\forall i : \sum_i^k \theta(CR_i, z_k) = 1$. Based on experimentation and reviewing literature such as [5], [12], we applied membership threshold $M_{TP} = 0.20$, resulting in each cr_q belonging to at least one z_k of the K topics, where $K = 20$.

Next, word topic membership vector φ_K is analyzed for determining topics. For each word in φ_k, if $\varphi(word, z_k)$ is greater or equal to $T_{TP} = 0.80$, corresponding word is added as the topic terms z_k. Thus each CR may be assigned to multiple topics (and associated terms). The information from these topics is combined with textual similarity data to make the prediction.

The algorithm takes as input all change requests (textual content and assigned topic terms), query CR text, query CR terms ($CRT_N, CR_q\, CR_{qT}$ respectively). Query CR represents the new request, for which file, effort, and duration needs to be predicted. We calculated the similarity between the query topic and each of the topics terms. If threshold T_M is satisfied, all files from CRs with the similar topic terms are added to I_{TM} (Steps 4 to 7). Effort and duration of similar CRs are added to the final list of impacted effort and duration, I_E and I_D respectively. File impact list is generated I_{BOW} using textual content similarity (same procedure as BOW). The final list of impacted files I_F is the intersection of I_{BOW} and I_{TM}. Thus, this technique generates results by considering both textual content similarity and underlying topic similarity.

3) BOW & File Coupling (BOW & CO): The third technique for change prediction takes into consideration the coupling information of files. Coupling has been used in some studies to support CIA [16]–[18]. We integrated the coupling information with Bag of words. Since this technique only additionally analyzes file information, it is not used for effort and duration prediction. The coupling information is generated using commit history stored in the code repository. We treated each commit as a separate transaction, and the files which appear together in a commit are considered to change together. From the commit analysis, a support matrix $S_{q \times q}$ is generated, where any matrix entry $S(a, b)$ is defined as:

$$S(a, b) = \mathcal{N}(F_a \cap F_b) \tag{2}$$

$\mathcal{N}(F_a \cap F_a)$ is a total number of times files F_a and F_b have appeared in the same commit, and q is the total number of files observed. The following is an example of a support matrix which shows how three files have changed. We can see that $F3$ has not changed with $F1$, changed two times with file $F2$ and finally, changed seven times by itself.

$$\mathbf{S} = \begin{array}{c} \\ F1 \\ F2 \\ F3 \end{array} \begin{array}{c} \begin{array}{ccc} F1 & F2 & F3 \end{array} \\ \begin{pmatrix} 2 & 2 & 0 \\ 2 & 4 & 6 \\ 0 & 2 & 7 \end{pmatrix} \end{array}$$

As using only support value introduces bias [16]–[18], a confidence matrix $C_{q\times q}$ is generated (by normalizing support values with total counts of change), where entry $C(a,b)$ is defined as:

$$C(a,b) = \frac{S(a,b)}{S(a,a)} \tag{3}$$

where $S(a,a)$ is the total number of times that file a has changed.

The procedure for generating the impacted file list in this technique is done by adding the following steps to Algorithm 1. For each file $f \in si_f$ (step 7), the coupled files are retrieved from confidence matrix $C_{q,q}$. If the corresponding confidence value for a coupled file is greater than T_{CO}, then it is added to the final list of impacted files I_F.

D. Evaluation Measures

For evaluating the performance of the proposed techniques, Recall, Precision and F-score are used. They are used measures in Information Retrieval studies [4].

- Precision P is the ratio of the number of correctly identified files to the total number of files (relevant and irrelevant) recommended i.e. retrieved files.

$$P(CR_q) = \frac{\#(\{relevant\ files\} \cap \{retrieved\ files\})}{\#\{retrieved\ files\}} \tag{4}$$

- Recall R is the ratio of the number of correctly identified files to the total number of relevant files.

$$R(CR_q) = \frac{\#(\{relevant\ files\} \cap \{retrieved\ files\})}{\#\{relevant\ files\}} \tag{5}$$

- *F-score* combines the precision and recall, where the weights of each depends on the value of β (equation 6). Traditionally $F1$ scores are mostly used (being the harmonic mean of precision and recall). However, studies such as [20], [13] suggest that Recall is more important than Precision. To reflect that, F2, F5 and F10 measures are used which assigns greater importance to recall.

$$F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{6}$$

- *Effort & Duration:* For duration and effort prediction, the evaluation needs to be performed differently. Instead of defining a fixed value as the target attribute, an error bound $E = 25\%$ is set. If the absolute difference between predicted and actual value (of effort and duration) is within 25% of the actual value, then the estimate is considered as a match. The company defined these tolerance values. From the list of predicted effort and duration I_E, I_D from algorithm 1,2, if any one of the values falls within E, it is considered a match for the corresponding issue for which the list is generated. The final evaluation is the percentage of queries (i.e., issues) for which effort and duration could be correctly predicted within bound E.

IV. Empirical Results

We used Leave-One-Out Cross Validation (LOOCV) to evaluate the performance of the model on predicting impacted files, effort and duration associated with each change request. In each iteration, one CR from data set plays the role of a test case, while the remaining change requests are used for prediction. The subject of the experiment are the two `Jira` projects. The corresponding file changes are captured from the respective `GitHub` repositories. If a file has been observed to change only once in the time span of analysis, then the corresponding file is omitted from analysis. The assumption is that, if a file changed only once in this duration, it is unlikely to change again in the future.

The value of T_B and T_{TM} has an inverse relationship to Precision and Recall. Based on repeated trials and error experiments, the value of T set to *0.3* presented with the best trade-off. Similarly, T_{CO} is set at 0.4 for both `Dental` and `Mail` projects respectively. From the results observed, the answer to the research questions defined earlier is stated below.

A. RQ1: Among the three approaches (i) Bag of words (BOW) (ii) Bag of words and topic modeling (BOW & TM) and (iii) Bag of words and file coupling (BOW & CO), which one works best better for predicting impacted files by a change request?

We did the experiments separately for each project. The performance of the three techniques for file impact prediction is shown in Table II and figures 3 - 5. In the charts, legend *D-* and *M-* denotes results of `Dental` and `Mail` projects respectively. The highest Recall of 0.67 and 0.56 for both `Mail` and `Dental` projects, was achieved with BOW & CO. This combination always generated the highest Recall values shown by Figure 3. Further considering F5, F10 values, BOW & CO also achieved the highest values (Figure 4, 5). Precision had the highest value of 0.12 with BOW. However, considering F1 and F2, the highest values (0.17 and 0.24 respectively)

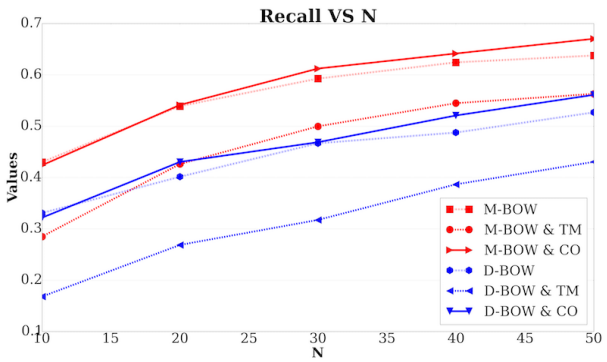

Fig. 3. File impact prediction result - Recall

TABLE II
ACCURACY MEASURE OF FILE CHANGE IMPACT ANALYSIS FOR THE THREE PROPOSED TECHNIQUES APPLIED TO THE MAIL AND DENTAL PROJECTS AND REPORTED FOR VARYING NUMBER N OF FILE FROM SIMILAR CHANGE REQUESTS.

Technique	N	MAIL						DENTAL					
		Precision	Recall	F1	F2	F5	F10	Precision	Recall	F1	F2	F5	F10
BOW	10	**0.11**	0.43	**0.17**	**0.27**	0.38	0.42	**0.12**	0.33	**0.17**	**0.24**	0.31	0.32
	20	0.07	0.54	0.12	0.23	0.43	0.51	0.08	0.40	0.13	0.22	0.35	0.39
	30	0.05	0.59	0.10	0.19	**0.43**	0.54	0.07	0.47	0.12	0.21	0.38	0.44
	40	0.04	0.62	0.08	0.17	0.41	0.55	0.06	0.49	0.10	0.19	0.38	0.45
	50	0.03	0.64	0.07	0.14	0.38	0.54	0.05	0.53	0.09	0.19	0.39	0.48
BOW & TM	10	0.07	0.28	0.12	0.18	0.26	0.28	0.07	0.17	0.10	0.13	0.16	0.17
	20	0.06	0.43	0.10	0.18	0.34	0.40	0.06	0.27	0.09	0.15	0.24	0.26
	30	0.04	0.50	0.08	0.16	0.36	0.45	0.05	0.32	0.08	0.15	0.26	0.30
	40	0.04	0.54	0.07	0.14	0.36	0.48	0.04	0.39	0.08	0.15	0.30	0.36
	50	0.03	0.56	0.06	0.13	0.34	0.48	0.04	0.43	0.08	0.15	0.32	0.39
BOW & CO	10	**0.11**	0.42	**0.17**	**0.27**	0.38	0.41	0.11	0.32	**0.17**	0.23	0.30	0.32
	20	0.07	0.54	0.13	0.24	**0.43**	0.51	0.09	0.43	0.14	**0.24**	0.37	0.41
	30	0.06	0.61	0.10	0.21	0.44	0.56	0.07	0.47	0.12	0.22	0.39	0.44
	40	0.05	0.64	0.08	0.18	0.43	0.57	0.06	0.52	0.11	0.21	0.41	0.49
	50	0.04	**0.67**	0.07	0.16	0.41	**0.58**	0.06	**0.56**	0.10	0.20	**0.42**	**0.52**

was observed for both BOW and BOW & CO. Although the precision is 1% in one configuration, it is better in terms of retrieving the number of impacted files. This is reflected in the higher Recall values (and subsequent F5 and F10 values).

The results indicate that file impact prediction can be improved if coupling information is added to the textual similarity measure. Textual similarity retrieves the important files in the initial stage. For each of these files, coupling further provides additional files which are likely to change with the original files. Coupling threshold restricts files with lower degree of association to enter the final list.

For the combination BOW & TM, the values are consistently lower across the board. On average the values are 6% to 7% less than the highest values obtained. This is because the grouping of CR is done primarily based on topic modeling, which has not been able to capture the context of the experiment correctly. The performance can most likely be improved if better context and meaning can be introduced with the topics.

Recall: Combining file coupling with Bag of Words retrieved the highest percentage (67%) of impacted files from the test cases.

In addition to the principal RQ1, two additional sub questions are formulated and described subsequently.

1) RQ1.1: How the techniques perform in dependence of the number of similar objects considered?: N controls the size of the prediction list. E.g., if $N = 10$, all evaluation metric will be calculated considering only the first ten files coming from the similar CRs. Increasing N positively impacts all the evaluation metric except Precision. If more results are considered for evaluation, then the size of the observed truth goes up. Despite increases in the number of matches (indicated by higher recall values) overall Precision values suffer. This holds for both `Mail` and `Dental` projects and also across all technique variations.

On the other hand, increasing N increases the Recall, which is reflected in all higher F-score values. The trend is reflected in both `Mail` and `Dental` projects. As increasing N leads to more results to be considered, this causes an increased number of matches. For all the techniques and dataset

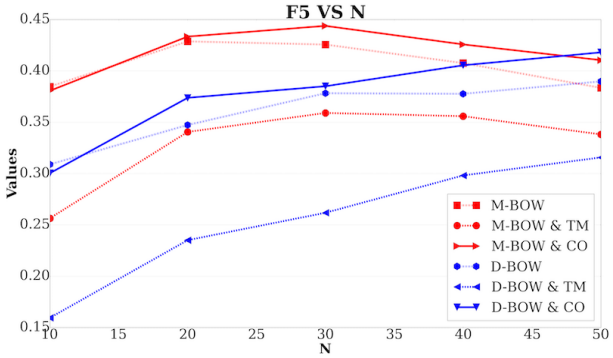

Fig. 4. File impact prediction result - F5

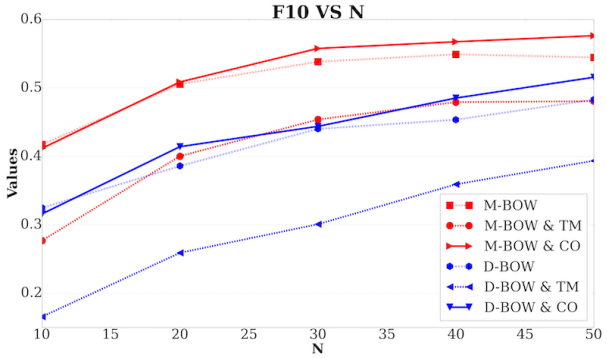

Fig. 5. File impact prediction result - F10

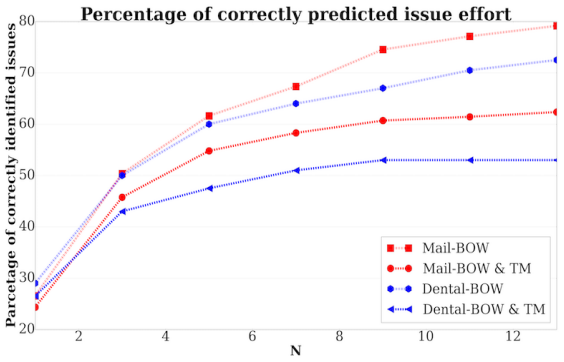

Fig. 6. Effort prediction results - Mail & Dental projects

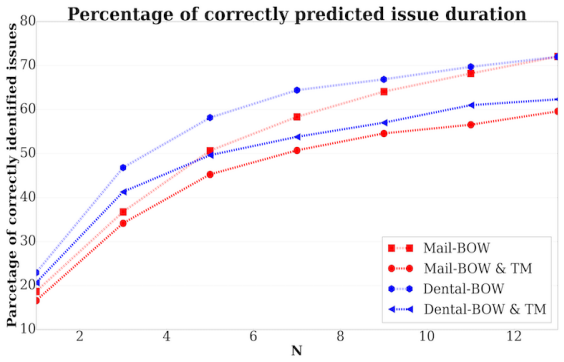

Fig. 7. Duration prediction results - Mail & Dental projects

combinations, there is a clear trend that increasing N increase number of impacted files observed. For increasing values of N, the inverse relationship between precision and recall is evident here, and trade-off needs to be made depending on the scenario. For different experiments, N has been varied between 10 and 50. For values between $N = 30 - 40$ the highest F5 and F10 values are observed. The Recall value is also high in this range of N. This indicates that files can be relatively well predicted by analyzing top 30-35 results.

Number of similar items(N): Increasing N positively impacts Recall and F-score, but negatively affects Precision.

2) RQ 1.2: How the techniques perform in dependence of the type of change request?: For determining the effect of change request type on the file impact results, the same experiment was run but with the change requests separated into *Bug* and *Non-Bug*. Figure 8 and 9 shows the Recall values obtained by separating *Bug* and *Non-Bug* issues. The red and blue line indicates the value of F10 obtained by considering *Bug* and other CR types separately.

For both `Mail` and `Dental` projects, Recall values are clearly higher for *Bug* change requests. This indicates Bug issues are handled with better diligence and the changes are more systematically logged into the system. Based on further examination it was found that most CR belonged to the type *Bug*. We found 62% and 68% of CR belonged to this category for `Dental` and `Mail` projects respectively.

Change Request Type: *Bug* type change requests account for more than 50% of total requests and file prediction results are better when only *Bug* type is used.

B. RQ2: How well can textual similarity as defined in RQ1 also be used for predicting effort and duration of implementing a change request?

Figure 6, 7, show the number of correctly predicted change requests with respect to duration and effort. The value of N specifies the number of top similar change requests considered when making duration and effort predictions. Considering duration, the percentage of correctly matched CR varies between 16% and 72% for `Mail` project. For the `Dental` project duration percentage ranges between 20% and 71%. Similar analysis for effort estimation reveals that percentage of correctly classified CR varies between 38% and 84% for `Mail` project, 24% and 79% for `Dental` projects. Considering the best-observed results, Bag of Words outperformed Topic model by at about 1% and 6% for duration and effort prediction respectively.

Duration: Just considering *3 most similar issues*, duration could be predicted for 50% of test cases.

With increasing N there is an upward shift in the prediction results for both effort and duration. Although experiments have been up to $N = 13$, reasonable results can be achieved by examining top 7 similar change requests (78% and 64% change requests can be correctly predicted with respect to effort and duration respectively)

Duration + Effort: Considering *Top 7 most similar issues* effort and duration could be predicted for 67% and 64% of the test cases respectively.

V. Literature Review

In the development domain, maintenance activities usually initiate with the first delivery of the software [11]. CIA is also referred as ripple effect analysis [2]. This chapter discusses some of the CIA literature.

Gethers et al. [16] propose an integrated approach to Impact Analysis consisting of three components. First, textual descriptions from change request are used to find similar CRs and software entities (i.e., methods, files, classes, etc.), for generating a ranked list of impacted methods. Second, the co-changed pattern is used for creating potential impact sets. Third, execution traces are made with respect to individual files or features, representing a hierarchy of executed methods. Kagdi et al. [18] combine conceptual and evolutionary techniques for CIA which supports file and method level granularity. Compared to [16], [18] is different in terms of exclusion of trace data.

TABLE III
PERFORMANCE OF DIFFERENT FILE IMPACT ANALYSIS TECHNIQUES IN LITERATURE ("-" INDICATES UNAVAILABLE NUMBERS).

	Cutoff/N	**10**		**20**		**30**		**40**		**50**		**Data**	**Number of projects**
Study		**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**		
BOW & CO	Min	0.11	0.32	0.07	0.43	0.06	0.47	0.05	0.52	0.5	0.32	Industry	2
	Max	0.11	0.42	0.09	0.54	0.07	0.61	0.06	0.64	0.6	0.42		
Borg et al. [8]	Min	-	-	-	-	-	-	-	-			Industry	1
	Max	0.04	40	0.03	40	0.02	40	0.01	40				
Gethers et al. [16]	Min	0.06	0.06	0.05	0.12	0.04	0.14	0.04	0.18	-	-	Open source	4
	Max	0.14	0.37	0.1	0.53	0.08	0.64	0.07	0.75	-	-		
Kagdi et al. [18]	Min	0.01	0.01	0	0.01	0	0.01	0	0.01	0	0.01	Open source	6
	Max	0.14	0.54	0.1	0.64	0.08	0.7	0.06	0.73	0.05	0.78		
Zanjani et al. [28]	Min	0.02	0.01	0.02	0.06	-	-	-	-	-	-	Open source	1
	Max	-	-	-	-	-	-	-	-	-	-		
Canfora et al. [10]	Min	0.05	0.2	-	-	-	-	-	-	-	-	Open source	3
	Max	0.2	0.4	-	-	-	-	-	-	-	-		
Torchiano et al. [26]	Average	0.18	0.23	-	-	-	-	-	-	-	-	Open Source	5
Zimmerman et al. [29]	Average	-	0.33	-	-	-	-	-	-	-	-	Open source	8

Zanjani et al. [28] combine interaction information (the file which has been interacted by developers but not committed) with version history to derive an IR model. Corpora is created using comments and identifiers from interacted and committed source files, commit messages and associated CR descriptions. Torchiano et al. [26] use an IR technique for change resolution. The proposed approach constructs corpora of file descriptors where each entity is described by a combination of comments in the source code and commit messages. The queries are generated using combinations of terms appearing in change request descriptions. Ranking of the files are generated based on the occurrence of query terms in the corresponding file comment or commit message data.

Table III show the results observed in literature. The first row shows the results from this paper. Only BOW & CO is shown as it had the best results among the three techniques investigated. In the table, only CIA technique which has been evaluated using industry data is Borg et al. [8]. However, the focus of the study was to uncover impact on non-code entities specifically (in contrast, the focus of this paper is to uncover impact on files, effort, and duration). All the other studies are shown in the table concentrate on CIA, where the objective is to identify the impact on files (on the underlying methods in the files). They have all been evaluated on open-source data. In Table III, *min* and *max* reflect the minimum and maximum value observed for any of the configuration/projects in the study. For some of the studies, all the data for different thresholds were not available (indicated by "-"). From the table, it is visible that there is a big difference in the maximum and minimum values. The observation is that the performance greatly varies with the project (data). The data analyzed in these studies share different properties, characteristics, and trend, as highlighted in the table. Despite differences with data, from analysis of Table III two key findings are highlighted. One, comparing results of BOW & CO with the open source studies, it is clear that the maximum values attained are very similar. Thus BOW & CO has similar predictive abilities in terms of file prediction. Two, as the value of N increases the precision falls and recall increases. This the same trend that we observed in our results.

Comparing with Literature: Evaluation values observed are comparable to those observed in literature, despite apparent differences in data and application.

VI. LESSONS LEARNED

Collaboration between industry and academia leads to many findings which are beneficial for both parties. This section shares the lessons learned both from industrial and academic perspective. The hope is that these findings might help future collaborations in similar settings.

Planning & Organization: Comparing to the manual approach, Vector impact (required effort, duration and file changes) of change requests make the task of allocating scarce resources less tedious and error-prone. Various development activities can be efficiently planned under time and resource constraints. Essential core-features can be better prioritized using vector estimations of changes. This is vital for any new or emerging company like Brightsquid which focus on MVPs.

Customer Satisfaction: For a given change request, compared to manual estimations, vector CIA will help to make more realistic release decisions. For future releases, this includes the setting of more realistic proposed feature-set and release dates. For ongoing maintenance activities, vector estimates will help to reduce problem fixing times, allowing for better conformance to service level agreement. All of these increase the likelihood of achieving customer satisfaction, making it easier for Brightsquid to maintain the market reputation and instill consumer confidence.

Forecasting Return on Investment: Using effort estimation data, Brightsquid can more accurately estimate expenses for different products. Also, duration estimation can help to forecast revenue streams. All of these culminate into a more predictable analysis of the financial growth and return on investment over time.

File co-change pattern is valuable: File co-change pattern is a vital source of information for predicting file changes. The information is simple to calculate and readily available.

The data can be extracted from any change interval and does not depend on the CR repository. The positive aspect of using coupling is reflected from the higher metric values for BOW & CO method.

Higher recall and lower precision values: Low precision values are valid not only for this CIA study but also true for any traceability research. Due to a multitude of reasons, the primary of which are the points discussed in this section and also in threats to validity. A vital factor is the size of impact list N. Increasing N increases the value of recall, as there will be a greater chance of detecting changes by generating a larger list; at the expense of added cognitive effort. However, due to a higher number of false positives, the overall value of precision goes down. A similar trend is also observed from the literature.

VII. Threats to Validity

This section discusses the threats to validity of the approach and underlying results, presented in this paper. The following threats have are highlighted.

Bias on CR type: There is a bias on the types of change requests logged into the issue management system. By analyzing both projects, most of the CR belong to one category, i.e., Bug, which accounted for 60% of the total CR numbers. For other types of CR, then there will be none or very few existing instances to compare. Thus the performance may be negatively impacted.

CR Id for traceability: For linking CR to file changed and creating the ground truth, CR Id is searched in commit messages, and subsequent linking takes place. However, sometimes developers mention other CR Ids which are not directly related to the change but are meant to be used as references. In such cases the generation of the ground truth becomes biased.

Coupling assumption: A strong assumption of this paper (as well literature utilizing coupling) is that if two files appear in the same commit, then are related, i.e., should be coupled. However, the actual existence of coupling depends on the circumstances of change. The presence of two or more files in the same commit does not always guarantee relation.

Few similar instance from the past: CIA utilize historical data for impact predictions, will not perform well if no similar changes are found from history. For example, if for a change request no or few previous requests are considered, or for a file, no coupled files are found. For both cases, the performance will be negatively impacted. This applies to the techniques of this paper as well any study which has utilized historical data for prediction.

Traceability is low: CIA methods which utilize past historical changes, rely heavily on traceability of the changes. Traceability helps to give meaning to code changes and use them for impact prediction. However, very few number of changes are traceable to the code. Roughly only about one-third of the changes are traceable for `Mail` and half changes are traceable for `Dental` project. Having higher traceability enhances performance and evaluation.

Traceability: Only about one-third of the change requests are traceable to the code changes(commits).

Frequency of file is not uniform: We also performed frequency analysis of the number of times files changed. It was found that only some of the files changed many times. Most of the file change only a few number of time. Thus when change needs to be predicted for a new CR, the files which change more frequently are more likely to show up in the sample results. This is one of the prime reasons for lower precision values.

File Change Frequency: A large number of files change only a few times.

VIII. Conclusions & Future Work

The ability to react and implement change requests is of pivotal importance for business success. No silver bullet technique can be expected to provide high accuracy in the prediction of

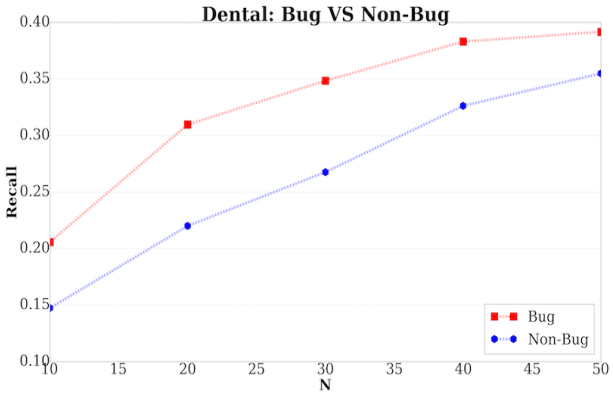

Fig. 8. Dental Bug VS Non-Bug issues - file impact prediction

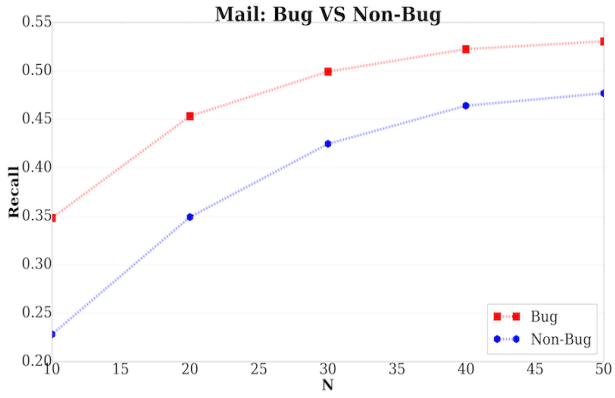

Fig. 9. Mail Bug VS Non-Bug issues - file impact prediction

impacted files. We could show that among existing methods, the combination of similarity-based search applying *Bag of Words* in conjunction with utilizing file coupling provides industrially acceptable solutions. Brightsquid considers the recommendations as a valuable input for their human decision-making about which and how many change request should be handled in which release.

The extension of the similarity-based reasoning on predicting not only impacted files, but also the duration, and effort needed to implement the change request, increasing the efficiency of the method and increases the industrial relevance of the approach. Both estimates so far were done ad hoc and manual, resulting in low accuracy. Using the new estimates almost without additional effort is attractive for Brightsquid's project management and supports their strategy of developing Minimum Viable Products.

Improving the overall prediction results is not exclusively seen as an algorithmic problem. Instead, it is considered being related to improving the content, structure and completeness of data collected. Also, additional CR attributes can be included in the analysis. Finally, the evaluation has been based on only the presence or absence of the target file. The position of the file on the list can be analyzed for evaluating how well the results appear near the top.

ACKNOWLEDGMENT

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada, Collaborative Research and Development Grants (NSERC-CRD-486636-2015) and by Alberta Innovates Technology Future (AITF).

REFERENCES

[1] M. Acharya and B. Robinson. Practical change impact analysis based on static program slicing for industrial software systems. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 746–755, New York, NY, USA, 2011. ACM.

[2] R. S. Arnold and S. A. Bohner. Impact analysis-towards a framework for comparison. In *Software Maintenance ,1993. CSM-93, Proceedings., Conference on*, pages 292–301, Sep 1993.

[3] I. R. M. Association. *Business Intelligence: Concepts, Methodologies, Tools, and Applications*. IGI Global, Hershey, PA, USA, 1st edition, 2015.

[4] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[5] A. Barua, S. W. Thomas, and A. E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Softw. Engg.*, 19(3):619–654, June 2014.

[6] A. Begel and T. Zimmermann. Analyze this! 145 questions for data scientists in software engineering. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 12–23, New York, NY, USA, 2014. ACM.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[8] M. Borg, K. Wnuk, B. Regnell, and P. Runeson. Supporting change impact analysis using a recommendation system: An industrial case study in a safety-critical context. *IEEE Transactions on Software Engineering*, PP(99):1–1, 2016.

[9] G. Canfora and L. Cerulo. Impact analysis by mining software and change request repositories. In *Proceedings of the 11th IEEE International Software Metrics Symposium*, METRICS '05, pages 29–, Washington, DC, USA, 2005. IEEE Computer Society.

[10] G. Canfora and L. Cerulo. Fine grained indexing of software repositories to support impact analysis. In *Proceedings of the 2006 International Workshop on Mining Software Repositories*, MSR '06, pages 105–111, New York, NY, USA, 2006. ACM.

[11] Y. C. Cavalcanti, P. A. da Mota Silveira Neto, I. d. C. Machado, T. F. Vale, E. S. de Almeida, and S. R. d. L. Meira. Challenges and opportunities for software change request repositories: a systematic mapping study. *Journal of Software: Evolution and Process*, 26(7):620–653, 2014.

[12] T.-H. Chen, S. W. Thomas, and A. E. Hassan. A survey on the use of topic models when mining software repositories. *Empirical Software Engineering*, 21(5):1843–1919, 2016.

[13] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou. Utilizing supporting evidence to improve dynamic requirements traceability. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 135–144, Aug 2005.

[14] R. Colomo-Palacios, C. Casado-Lumbreras, P. Soto-Acosta, F. J. García-Peñalvo, and E. Tovar. Project managers in global software development teams: A study of the effects on productivity and performance. *Software Quality Journal*, 22(1):3–19, Mar. 2014.

[15] A. N. Duc and P. Abrahamsson. *Minimum Viable Product or Multiple Facet Product? The Role of MVP in Software Startups*, pages 118–130. Springer International Publishing, Cham, 2016.

[16] M. Gethers, B. Dit, H. Kagdi, and D. Poshyvanyk. Integrated impact analysis for managing software changes. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 430–440, Piscataway, NJ, USA, 2012. IEEE Press.

[17] M.-A. Jashki, R. Zafarani, and E. Bagheri. Towards a more efficient static software change impact analysis method. In *Proceedings of the 8th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*, PASTE '08, pages 84–90, New York, NY, USA, 2008. ACM.

[18] H. Kagdi, M. Gethers, D. Poshyvanyk, and M. L. Collard. Blending conceptual and evolutionary couplings to support change impact analysis in source code. In *2010 17th Working Conference on Reverse Engineering*, pages 119–128, Oct 2010.

[19] S. L. Lim and A. Finkelstein. *Anticipating Change in Requirements Engineering*, pages 17–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[20] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora. Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Trans. Softw. Eng. Methodol.*, 16(4), Sept. 2007.

[21] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[22] S. McGee and D. Greer. Software requirements change taxonomy: Evaluation by case study. In *Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference*, RE '11, pages 25–34, Washington, DC, USA, 2011. IEEE Computer Society.

[23] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[24] E. Ries. How today's entrepreneurs use continuous innovation to create radically successful businesses, The lean startup, 2011.

[25] J. Śliwerski, T. Zimmermann, and A. Zeller. When do changes induce fixes? *SIGSOFT Softw. Eng. Notes*, 30(4):1–5, May 2005.

[26] M. Torchiano and F. Ricca. Impact analysis by means of unstructured knowledge in the context of bug repositories. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, pages 47:1–47:4, New York, NY, USA, 2010. ACM.

[27] B. J. Williams, J. Carver, and R. B. Vaughn. Change risk assessment: Understanding risks involved in changing software requirements. In *Software Engineering Research and Practice*, pages 966–971, 2006.

[28] M. B. Zanjani, G. Swartzendruber, and H. Kagdi. Impact analysis of change requests on source code based on interaction and commit histories. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 162–171, New York, NY, USA, 2014. ACM.

[29] T. Zimmermann, P. Weisgerber, S. Diehl, and A. Zeller. Mining version histories to guide software changes. In *Proceedings of the 26th International Conference on Software Engineering*, ICSE '04, pages 563–572, Washington, DC, USA, 2004. IEEE Computer Society.