2017 International Conference on Applied Mechanics and Mechanical Automation (AMMA 2017)
ISBN: 978-1-60595-471-4

FACIA: A Fully Automatic Change Impact Analysis Method for Large Scale Requirements

Wan-yu CHEN*, Hong-hui CHEN, Tao CHEN and Fei CAI

Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
Changsha, China

Keywords: Blooming changes, Change impact analysis, Impact units, Fully automatic technology.

Abstract. The blooming change of requirements poses a challenge for change impact analysis (CIA), especially in large scale software development. Existing CIA techniques exploiting manual analysis and automatic methods suffer from high cost, low accuracy and expert dependence in current industrial practice. Therefore, this paper proposes a fully automatic change impact analysis (FACIA) method to overcome the aforementioned shortcomings. We consider the requirement changes generally happen as the form of phrases. The impact units for the changed phrases may contain the changed phrases (CP1), the co-occurrence phrases of CP1(CP2), the tokens of CP1 in the changed requirement, the similar phrases of CP1 and the similar phrases of CP2 in the whole requirements. Then five algorithms are designed based on the combinations of those impact units, so as to get a more precise change propagation and generate the lists of change impact automatically. We conduct extensive evaluations for the proposed approach with two different industrial data sets. The results show that compared with the method of expert depended, our approach can get a reliable sorted list for CIA more quickly with lower cost.

Introduction

The need for an efficient change impact analysis (CIA) method with low consumption is increasing with the blooming and frequent changes in requirements of software engineering. When changes in requirements are proposed, the impact of these changes on other requirements, design elements and source code needs to be analyzed in order to determine parts of the software system to be changed. In this paper, we focus on change impact analysis in requirements only.

Many studies, such as [1, 5-7, 11-13], perform CIAs manually, and thus, the analyses are time consuming, labor-intensive and prone to human error. Without information that precisely determines how a change propagate through the relations, the requirement traceability(RT)-assisted technique in CIAs [11] covers all related requirements as impacted results, which contains error-impacted requirement sets and often misses various true impact sets. Another type of semiautomatic method is not well-suited for massive changes, and it improves the accuracy of the results when experts provide the change propagation [1]. The accuracy of this approach depends on the precision of the change propagation provided by experts.

In this paper, a fully automatic change impact analysis (FACIA) method is proposed to replace the RT assistant and to be applied in vast and frequent change appearances especially when lack of experts. In our method, we analyze the change propagation that composed of several impact units with specific weights. For a certain change, the following two direct impact units are considered: the changed phrase and phrases with high co-existence with the changed phrase (called the emphasized phrase). To improve the precision of the change propagation, five impact units are extended based on the two direct impact units, and five algorithms are designed according to various combinations of the impact units. Meanwhile, phrases in the two basic impact units have a high probability of existence in the propagation provided by experts, so this method is valid to guarantee the recall of the model without experts. We use correct impact sets for each change provided by experts as the ground truth and compare the results among our method and several other approaches. It proves our method can achieve a high accuracy and avoid noise phrases.

Contribution: The test and analysis results suggest that the contributions of the proposed approach are as follows:

1. FACIA enables an accurate CIA, especially when frequent changes occur in the extensive requirements. It can reduce the manpower and time requirements and avoid human-caused errors.

2. It can improve the accuracy of the CIA results without experts.

3. It can be used as a coarse-grained evaluation of CIA without expert assistance, as it provides a candidate result with relatively high accuracy.

Our Approach

Overview

The proposed approach can be divided into three steps:

Step 1: Phrase detection. The POS Tagger module and chunker module are used in NLP toolkit to complete the phrase annotations and phrase detection job. The POS Tagger module labels the tokens with their speech. Then, it extracts the phrase structure with the regular expressions in the module. The Chunker module is used to divide those phrases into tokens and discard the stop words. The output of Step 1 is the total phrases of the requirements.

Step 2: Calculating the similarity between phrases with the WORDNET technique. WORDNET can determine the similarity between tokens [4]. An algorithm is used to calculate the similarity between requirements and phrases [1]. This algorithm is modified and applied in our approach to calculate the similarity between phrases. The output of Step 2 is the similarity among all phrases with every other phrase. The first two steps are executed one time for a certain requirement context in CIA.

Step 3: Analyzing the impact units and receiving the sorted impact list. This method is a fully automatic method of finding the sorted impact list, thus, expert analysis is not required. As a result, the impact units must be analyzed before designing the algorithms. Here, five impact units are considered: the change phrase unit (L1), the similarity of the L1 unit (L2), the co-existence with the change phrase unit (L3), the similarity of the L3 unit (L4) and the tokens contained in the change phrases unit (L5). Then, with different impact unit combinations, five candidate algorithms are designed to calculate the impact factor for each requirement. The algorithms are then tested with two case studies. Finally, the best algorithm is obtained from the results. This step can provide a sorted list of the impacted requirements for a certain change.

Calculation of the Phrase Similarity

First, we calculate the similarity between two tokens based on their speech labels and meanings with the package Word-Net: Similarity. Then we obtain the similarity between phrases with the following process including two steps..

Step 1: The similarity matrix for *phrase1* and *phrase2*. For *phrase1* and *phrase2*, $<t_{11}, t_{12}, ..., t_{1n}> \in phrase1$, $<t_{21}, t_{22}, ..., t_{2m}> \in phrase2$. We calculate the similarity between the tokens in these two phrases and obtain the similarity matrix $S_{<p1,p2>}$ in Eq. 1:

$$S_{<p1,p2>} = \begin{bmatrix} s_{11} & \cdots & s_{1m} \\ \vdots & \ddots & \vdots \\ s_{n1} & \cdots & s_{nm} \end{bmatrix} \tag{1}$$

n is the number of tokens in *phrase1*, m is the number of tokens in *phrase2*, and s_{ij} indicates the similarity between token i in *phrase1* and token j in *phrase2*.

Step 2: Calculating the similarity between *phrase1* and *phrase2*. The algorithm proposed in [1,10] is used to calculate the similarity between a phrase p and requirement R. Eq. 2 shows this algorithm:

$$S = 2 \times \frac{\text{sum of the weights of edges in optimal matching}}{N_1 + N_2 + 0.5 \times (N_3 - 1)} \tag{2}$$

The numerator is the sum of the optimal similarity between every token of p and those of R. Here, N_1 is the number of tokens in phrase p, N_2 is the number of matched tokens from R, and N_3 is the number of phrases from R involved in the optimal matching.

This algorithm is modified and applied to calculate the similarity between phrases in our approach. Thus, N_1 is the number of tokens in phrase1. N_2 is the number of tokens from phrase2 because they are all matched. Only one phrase (phrase2) is involved in the optimal matching, as $N_3=1$ in this approach. Then, S_1 is set to be the similarity of *phrase1* to *phrase2*. S_2 is the similarity of *phrase2* to *phrase1*. The average is the similarity S between them. Thus, the calculation process is as follows:

$$g(t) = \arg\max(sim(\mathrm{t}, \mathrm{t}')) \tag{3}$$

In Eq. 3, t is the token in *phrase1*, whereas t' is the token in *phrase2*. g(t) is the t' in *phrase2* that has the maximum similarity with the token t in *phrase1*. So,

$$S = \frac{S_1 + S_2}{2} = \frac{\sum_{t \in phrase1} sim(\mathrm{t}, \mathrm{g(t)}) + \sum_{t \in phrase2} sim(\mathrm{t}, \mathrm{g(t)})}{n + m} \tag{4}$$

Change Impact Analysis

Change Impact Sets. when one requirement gets changed, we want to know which requirements are impacted in the requirements context through CIA. Here we give a definition of the concept of the change impact unit: A set contains the phrases that is related to the changed phrases directly or indirectly.

The impact of a certain change on a requirement mainly depends on whether the requirement contains change impact units. Thus, if a requirement contains the phrases in the impact units, this requirement will be impacted to some extent. Here, the impact factor $IF[R]$ is used to quantify this impact. In our approach, we first define the following impact units and use an effective combination of the impact units to express the change propagation and then calculate the impact factor for each requirement.

The approach in this work is fully automated, with no expert analysis. Thus, the impact units are proposed before designation of the calculation algorithm. The following are the impact units considered in our approach:

(1) L1: Change phrase;

(2) L2: Similarity set of L1;

(3) L3: A phrase that exists in the same requirement as the change phrase. For L3 set, the co-existence frequency with the change phrase is also considered. It is obviously that when two phrases have higher co-existences in the requirement context, they have a stronger relationship.

(4) L4: Similarity set of L3;

(5) L5: Tokens that exist in the change phrase.

Here we give it a definition of the weight of impact units: A requirement is impacted by different impact units to different extents. It is obviously that the direct impact unit brings more influence than the indirects on the requirement which contains both of them. Therefore, a parameter $W[\mathit{impact\ unit}]$ is used to describe the weight for an impact unit. In our approach, we set the weights of the impact units as $W = [2,1,1,0.5,2]$, which is also used in our experiment.

Algorithms for Calculating the Impact Factor. Five algorithms are designed for impact factor calculation based on different impact unit combinations. The output array $IF[R]$ of the algorithms is the impact factor for the requirement R in the sorted list. Then, the impact factor for requirement R is calculated by Eq. 5.

$$IF[R] = \sum_{L(\mathrm{impact})} \sum_{L(R)} S(\mathrm{p}_1, p_2) \times W[\mathit{impact\ unit}] \tag{5}$$

$L(R)$ is the set of phrases contained in requirement R. $L(\mathrm{impact})$ is the set of phrases contained in the impact unit. L is the set of phrases contained in the total requirements. $S(p_1, p_2)$ is the similarity between *phrase1* and *phrase2*. *Phrase1* belongs to $L(R)$, whereas *phrase2* belongs to $L(\mathrm{impact})$.

As the five impact units effect the change impact analysis to different extents, algorithms are designed according to various impact unit combinations. Algorithm 1 considers the L1 and L2 impact units into the change propagation, whereas Algorithm 2 considers L1, L2 and L4. The impact factor of each algorithm can be calculated by Eq. 5. We traverse all the requirements and once the requirement contains the phrase in the impact unit, the impact factor of the requirement is added by the product of the impact unit' weight and the phrase similarity.

In Algorithm 3, the impact factor contains the L1, L2 and L3 impact units into the change propagation. Here, we also add the co-occurrence between a phrase with the change phrase into Eq. 5 as another weight for L3. Algorithm 4 adds the impact of the L5 set based on Algorithm 2 into the change propagation. The existence of phrases in L5 is considered. Thus, when a requirement contains the tokens in L5, the impact factor is increased. Algorithm 5 adds the impact of the L5 set based on Algorithm 3 into the change propagation. Those five algorithms represent for different expresses of the change propagation. In next section, we evaluate the performance of the five algorithms respectively with two industrial cases.

Evaluation

Dataset

In this section, the accuracy and use of the proposed approach are investigated through two industrial case studies. The first case examines a 3G mobile service platform. The case is based on a real system and is written by practitioners [1]. The second case studies an electronic tourist guide application developed by final-year students at the University of Salerno (Italy) [9]. First, Step 1 is used to obtain the phrases of the requirements. Then, three experts are invited to provide the correct impact sets for each change as the proving ground of the proposed method. Table 1 provides the information for the two case studies:

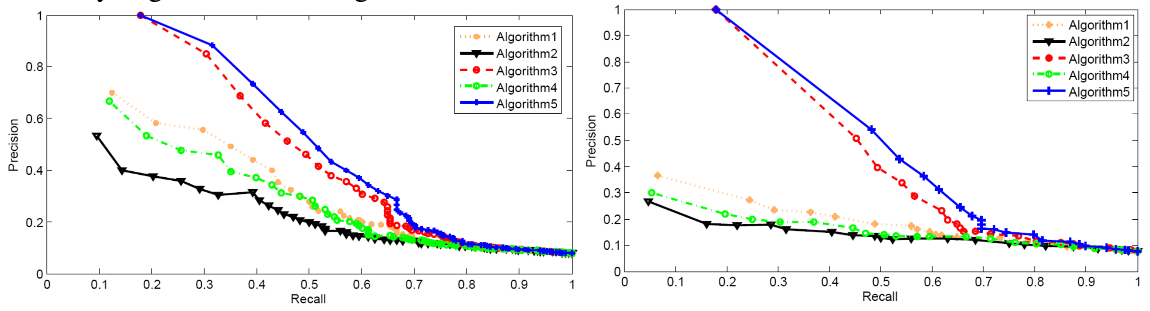
Table 1. Cases studies used in the evaluation.

Cases	requirements	Phrase detection	tokens	Change phrases
3G mobile service	72	206	263	30
Electronic tourist guide application	22	65	90	15

Experiments

Question 1: Which algorithm is more effective and yields the best results?

The five algorithms are tested with two cases. Fig. 1(a) shows the recall-precision lines for the five algorithms for case 1. Fig. 1(b) illustrates the five algorithms for case 2. The horizontal axis shows the recall for every cutoff, whereas the vertical axis shows the precision for every cutoff. Fig. 1(a) and Fig. 1(b) indicate that Algorithm 5 yields a better result than the other four algorithms, particularly Algorithm 2 and Algorithm 4.



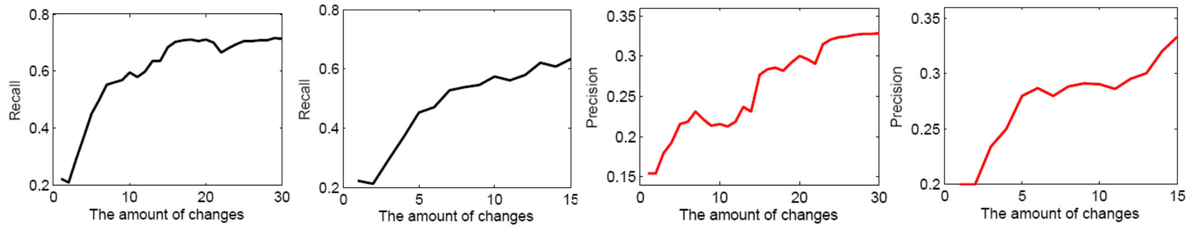
(a) Recall-Precision of the five algorithms in case1. (b) Recall-Precision of the five algorithms in case2

Figure 1. Recall-precision of the five algorithms.

Compared with the other algorithms, the impact units include the following: L1, L2, L3 and L5 in Algorithm 5 without impact unit L4 included in Algorithm 2 and Algorithm 4. This recommended combination demonstrate that the direct impact units have a positive influence on the accuracy of the results in the automatic approach. However, the similarity set of L3, namely, the L4 impact unit could be considered in the method with expert analysis and may provide a positive effect. When it is introduced into the automatic approach, the level of noises will increase and various actual impacted requirements will be neglected.

Question 2: What are the advantages of the proposed approach?

Accuracy. First, the Recall and Precision targets of the proposed approach are evaluated. The cutoff point is set according to the method proposed by reference [1], which is computed automatically and can thus be directly obtained with computer calculations. The amounts of changes gradually increase and are tested for those two targets. Fig. 2 shows the Recall and Precision of the proposed Algorithm 5 with increasing changes in the two case studies.



(a) Recall for case1. (b) Recall for case2. (c) Precision for case1. (d) Precision for case2.

Figure 2. Recall and precision of our algorithm 5 with the increase of changes.

Fig. 2 illustrate that the proposed approach has an improving trend of recall and precision with the increase of changes in the requirements. It illustrates that the approach is well suited for the large-scale changes appearance in the context of requirements with a reliable accuracy. These results present a key advantage in applying the automatic approach within a change-blooming situation.

Scalability. To evaluate the scalability of the proposed approach, the execution time of the approach proposed by reference [1] (NLP) was compared to that of our proposed approach (FACIA). Table 2 provides the execution times for the main computational tasks of case 1 when applying NLP and FACIA. The execution times were measured on a laptop with a 2.3GHz CPU and 8GB of memory.

Table 2. The comparison of execution time.

Steps	NLP	FACIA
Phrase detection	12s	15s
Syntactic similarity	8s	-
Semantic similarity	74s	90s
Sorted list generation(average)	9s	0.041s
sorted list generation(worst case)	11s	0.055s

In Table 2, the phrase detection step and similarity calculation occur once each. Table 2 illustrates the entire execution time of our proposed approach as it is fully computational. However, the execution time for the NLP approach shown in Table 2 does not contain the cost of the expert analysis process time in the approach. The results show that when faced with large amounts of changes in the requirement context, our proposed approach FACIA can provide the candidate sorted list for change impacts more rapidly, which is rather applicable in large scale software engineering.

Conclusions

In this paper, we present a fully automatic change impact analysis approach with NL requirements. We analyze the change propagation with different combinations of change impact units and design five algorithms to get the change impact list of requirements. The evaluation results with two industrial cases demonstrate that our approach can provide a precise change impact analysis without experts, especially when more requirements changes happen. The main advantages of this approach is that we can provide a credible sorted list for the CIA rapidly at lower cost and is thus applicable in cases with extensive changes.

In the future, the following steps can be taken to enhance the accuracy of this fully automated approach. First, a more reasonable similarity calculation method can be used with a domain-specific knowledge source. An external source of knowledge, such as WORDNET, will add noise to the change impact results [2, 3, 8]. Second, the details in the algorithm can be modified to make it applicable to various change types. Finally, additional usability studies will be conducted to further validate the relation between change propagation and the emphasized phrases.

References

[1] Arora, C., Sabetzadeh, M., Goknil, A., Briand, L.C., Zimmer, F, Change impact analysis for natural language requirements: an nlp approach. In: 23rd IEEE International Requirements Engineering Conference. pp. 24-28. Ottawa, Canada (August 2015).

[2] Dahlstedt, A., Persson, A, Requirements inter-dependencies: state of the art and future challenges. In: Engineering and Managing Software Requirements. pp. 95-116. Springer, Berlin (August 2005).

[3] Falessi, D., Cantone, G., Canfora, G, Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. IEEE Trans Softw Eng 39, 18-44 (January 2013).

[4] Falleri, J., Huchard, M., Lafourcade, M., Nebut, C., Prince, V., Dao, M, Automatic extraction of a wordnet-like identifier network from software. In: International Conference on Program Comprehension. pp. 4-13. Braga, Portugal (July 2010).

[5] Goknil, A., Kurtev, I., van den Berg, K, Spijkerman, W, Change impact analysis for requirements: a meta-modeling approach. Inf Softw Technol 56, 950-972 (August 2014).

[6] Hata, M., Homae, F., Hagiwara, H, Semantic relatedness between words in each individual brain: an event-related potential study. Neurosci Lett 2, 72-77 (January 2009).

[7] Jurafsky, D., Martin, J, Speech and language processing: an introduction to natural language processing. Prentice Hall, New Jersey, USA (2000).

[8] Mahmoud, A., Niu, N, Using semantics-enabled information retrieval in requirements tracing: an ongoing experimental investigation. In: Annual Computer Software and Applications Conference. pp. 246-247. Seoul, Korea (July 2010).

[9] Mahmoud, A., Niu, N, On the role of semantics in automated requirements tracing. Requir Eng 20, 281-300 (January 2015).

[10]Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., Zave, P, Matching and merging of variant feature specifications. IEEE Trans Softw Eng 38, 1355-1375 (December 2012).

[11]Sunil, N., Jose, L., Sagar, S, A review of traceability research at the requirements engineering conference. In: 21st IEEE International Requirements Engineering Conference. pp. 222-229.

[12]Yang, H.B., Liu, Z.H., Ma, Z.H, An algorithm for evaluating impact of requirement change. J Inf Comput Sci 2, 48-54 (August 2007).

[13]Zhang, H., Li, J., Zhu, L., Jeffery, D., Liu, Y., Wang, Q., Li, M, Investigating dependencies in software requirements for change propagation analysis. Inf Softw Technol 56, 40-53 (Sep 2014).