Keeping Evolving Requirements and Acceptance Tests Aligned with Automatically Generated Guidance

Hotomski, Sofija ; Ben Charrada, Eya ; Glinz, Martin

Abstract: [Context and motivation] When a software-based system evolves, its requirements continuously change. This affects the acceptance tests, which must be adapted accordingly in order to maintain the quality of the evolving system. [Question/problem] In practice, requirements and acceptance test documents are not always aligned with each other, nor with the actual system behavior. Such inconsistencies may introduce software quality problems, unintended costs and project delays. [Principal ideas/results] To keep evolving requirements and their associated acceptance tests aligned, we are developing an approach called GuideGen that automatically generates guidance in natural language on how to modify impacted acceptance tests when a requirement is changed. We evaluated GuideGen using real-world data from three companies. For 262 non-trivial changes of requirements, we generated guidance on how to change the affected acceptance tests and evaluated the quality of this guidance with seven experts. The correctness of the guidance produced by our approach ranged between 67 and 89 percent of all changes for the three evaluated data sets. We further found that our approach performed better for agile requirements than for traditional ones. [Contribution] Our approach facilitates the alignment of acceptance tests with the actual requirements and also improves the communication between requirements engineers and testers.

DOI: https://doi.org/10.1007/978-3-319-77243-1_15

Originally published at:
Hotomski, Sofija; Ben Charrada, Eya; Glinz, Martin (2018). Keeping Evolving Requirements and Acceptance Tests Aligned with Automatically Generated Guidance. In: 24th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2018), Utrecht, 20 March 2018 - 22 March 2018, 247-264.
DOI: https://doi.org/10.1007/978-3-319-77243-1_15

Keeping Evolving Requirements and Acceptance Tests Aligned with Automatically Generated Guidance

Sofija Hotomski, Eya Ben Charrada, and Martin Glinz

Department of Informatics, University of Zurich, Zurich, Switzerland
{hotomski,charrada,glinz}@ifi.uzh.ch

Abstract. [**Context and motivation**] When a software-based system evolves, its requirements continuously change. This affects the acceptance tests, which must be adapted accordingly in order to maintain the quality of the evolving system. [**Question/problem**] In practice, requirements and acceptance test documents are not always aligned with each other, nor with the actual system behavior. Such inconsistencies may introduce software quality problems, unintended costs and project delays. [**Principal ideas/results**] To keep evolving requirements and their associated acceptance tests aligned, we are developing an approach called GuideGen that automatically generates guidance in natural language on how to modify impacted acceptance tests when a requirement is changed. We evaluated GuideGen using real-world data from three companies. For 262 non-trivial changes of requirements, we generated guidance on how to change the affected acceptance tests and evaluated the quality of this guidance with seven experts. The correctness of the guidance produced by our approach ranged between 67 and 89% of all changes for the three evaluated data sets. We further found that our approach performed better for agile requirements than for traditional ones. [**Contribution**] Our approach facilitates the alignment of acceptance tests with the actual requirements and also improves the communication between requirements engineers and testers.

1 Introduction

When developing or evolving systems, requirements constantly change and, in most cases, these changes affect other documentation artifacts. In practice, however, impacted artifacts too often are not kept aligned with changing requirements. To a significant extent, this is due to the additional effort required and to insufficient communication of requirement changes [1,2]. Losing the alignment between requirements and other documentation artifacts increases the risk of discovering mismatches between stakeholders' expectations and the actual software behavior only late, leading to unintended costs, delivery delays and unsatisfied customers. For example, when acceptance tests are not kept aligned with changed requirements, testers will report bugs for actual features that were introduced in a change.

24th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2018), Utrecht, The Netherlands. pp 247–264. In LNCS 10753, Springer.

In order to keep software documentation aligned and up-to-date when a system evolves, many researchers try to automatically identify which documents are related to each other and which of them are *impacted* by a change [3,4]. However, there is little research about how to actually *update* impacted documents, although it would be beneficial to have guidance about what actions to perform [5].

In our work, we contribute an approach for keeping acceptance tests aligned with evolving requirements, called GuideGen. GuideGen automatically generates guidance on how to modify impacted acceptance tests when requirements change. We take advantage of the fact that requirements and acceptance tests have much in common: both are usually written in natural language and contain information about what the system under development is expected to do: requirements specify what should be implemented [6] and acceptance tests validate whether the implementation satisfies the requirements of the stakeholders [7]. Due to this similarity, tracing from requirements to acceptance tests is not difficult. Our approach assumes that traces between every requirement and its associated acceptance test(s) exist. If this is not the case, automated trace generation techniques [4], [8] may be used for establishing such traces.

By analyzing changed sentences and words in a requirement, we derive guidance in form of a set of concrete suggestions about what should be changed in the acceptance test(s) associated with a changed requirement. Our tool also provides an easy way for communicating changes and the generated guidance to all interested parties. GuideGen aims at both reducing the effort for aligning acceptance tests with the actual requirements and improving the communication between requirements engineers and developers/testers.

In a previous paper [9] we presented the principal ideas of our approach together with some examples and a preliminary evaluation. In this paper we describe our method and the algorithms used in detail, give an overview of the GuideGen tool, and present the results of a thorough evaluation with real-world data.

The paper is organized as follows. In the next section we present our approach and its technical components. We then present our prototype tool in Sect. 3. Section 4 describes our evaluation. We discuss our results in Sect. 5. Related work is discussed in Sect. 6. Section 7 concludes the paper with a summary and outlook.

2 Our Approach

The goal of GuideGen is to identify all *relevant changes* in requirements that require the associated acceptance tests to be adapted and to *generate guidance* in natural language on how to adapt the acceptance tests based on these changes. An overview of our approach is shown in Fig. 1.

As soon as a requirements engineer applies changes to a requirement and saves them, our approach performs the following steps:

1. Identifying relevant change patterns: by comparing the old and the new version of the changed requirement we identify the elements that have been changed and their change types,

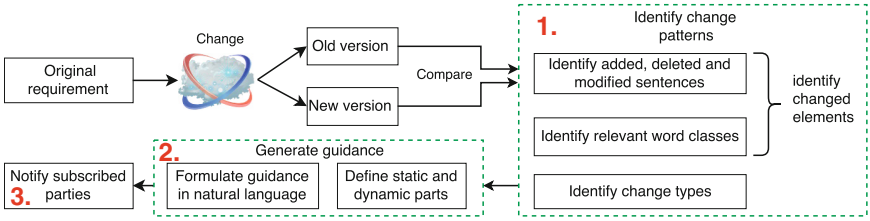

Fig. 1. Overview of the GuideGen approach

2. Generating guidance: in this step, we formulate suggestions in natural language on how to manage the changes,

3. Notifying subscribed parties: finally, the generated guidance and the changes can be communicated to the interested parties via e-mail.

In the remainder of this section, we present each of these steps in more detail.

2.1 Identifying Relevant Change Patterns

The goal of this step is to identify relevant patterns in the changes that are applied to a requirement. A *change pattern* is characterized by the change type (add, delete, or modify) and the changed element (a whole sentence or a word). If the changed element is a whole sentence, the change pattern is "Sentence is added" or "Sentence is deleted". If the changed element is a word, an example of a change pattern is "verb is deleted". *Relevant* change patterns are the ones whose changes require the acceptance tests to be adapted. In particular, relevant change patterns in our approach are the ones that directly or indirectly cause the change of some action, since acceptance tests contain a list of actions to be performed.

To identify the relevant change patterns, we first analyze the changes at a sentence level. Then we proceed by analyzing changes at a word level. Finally we classify each of the detected changes as relevant or irrelevant.

Analyzing changes at sentence level. In order to identify whether a whole sentence has been added, deleted or modified, we first split the old and the new version of the requirement into sentences using an implementation of the Stanford sentence splitting algorithm [10]. We get the list of old sentences (oldReq in further text) and the list of new sentences (newReq). Additionally, our tool transforms enumerated sentences into plain sentences. A plain sentence is a sentence without bullet points. An enumerated sentence contains the main part and at least two bullet points, e.g.

"A user can insert: - name,
- surname".

The sentence is transformed into: "A user can insert name" and "A user can insert surname". If a bullet point is added or deleted, the change is treated as an addition or deletion of a plain sentence. For instance, if we add "- e-mail", this change is treated as the addition of the sentence "A user can insert e-mail".

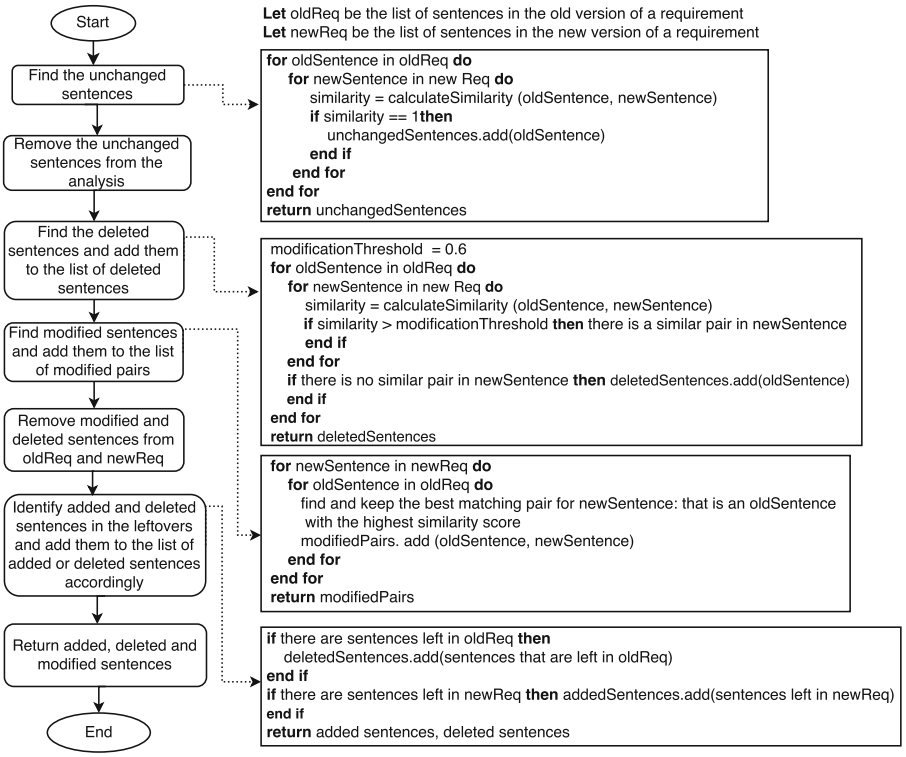

Fig. 2. The algorithm for identifying added, deleted and modified sentences

Otherwise, the addition of a noun that has no related verbs would be classified as an irrelevant change pattern.

We then compare all the sentences from oldReq with the sentences from newReq by calculating the similarity between them. Based on the similarity, we determine whether the sentence is unchanged, added, deleted or modified. The similarity is calculated using an existing semantic similarity toolkit [11]. In particular, we use greedy matching for word to word similarity that is based on WordNet. A flow diagram and the corresponding pseudo code of the algorithm are shown in Fig. 2.

If the similarity between a sentence in oldReq and one in newReq is equal to one, that sentence is considered to be unchanged. If a sentence in oldReq does not have a corresponding one in the newReq so that the similarity score between them is greater than the modification threshold of 0.6[1], then this sentence is deleted. When the similarity score between sentences is above the modification threshold, these sentences are candidates for modified sentences. We choose the best match – a pair of sentences whose similarity score is the highest among

[1] This is a heuristic value which yielded excellent performance in our evaluation, cf. Sect. 4.2.

other pair candidates. When we remove best matches, unchanged sentences and already identified deleted sentences from the oldReq and the newReq, there might be leftovers. The leftovers in newReq are added sentences and the leftovers in oldReq are deleted sentences. We illustrate this using the following example:

> A user can add new users to the group. ~~The addition of a new user must be first approved by the admin.~~ The admin and the user can modify personal data ~~and the status~~ of that ~~a~~ user. Only user can modify its status. The admin must be logged-in in order to modify personal data of a user.

Added words are green and underlined, removed words are red and struck through, while black words are unchanged.

Figure 3 shows the calculated similarities between the old and the new version of the changed requirement.

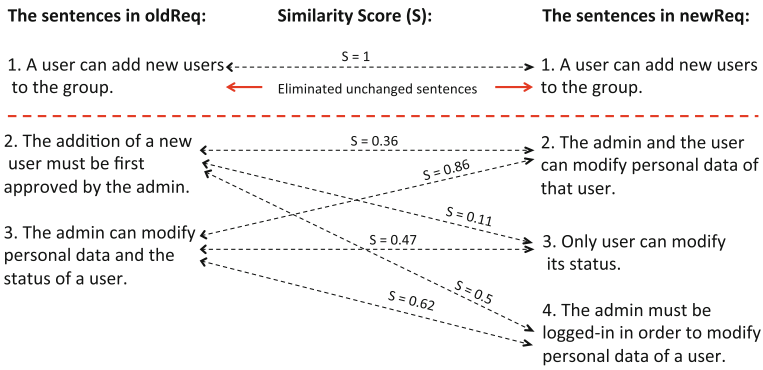

Fig. 3. Calculated similarity scores for the sentences in the example

The first sentence is eliminated from the further analysis because the similarity score is S(1, 1) = 1. Since all scores calculated for the second sentence, S(2, 2) = 0.36, S(2, 3) = 0.11 and S(2, 4) = 0.5, are below the modification threshold (0.6), the second sentence in the oldReq is found to be deleted. We defined the modification threshold based on experimentation: we calibrated it to the value that yielded the best results for identifying added, deleted and modified sentences. For the third sentence in old-Req we see that there are two matching sentences in the newReq so that the similarity is above the modification threshold: S(3, 2) = 0.86 and S(3, 4) = 0.62. We choose the best match in this case, i.e., S(3, 2). Therefore, the third sentence in the oldReq is modified to the second sentence in the newReq. The third and the fourth sentence in the newReq become leftovers. Since they are both in the newReq we find that these two sentences have been added.

ID	TEXT	WORD CLASS	DEPENDENCY NUMBER	GRAMMATICAL FUNCTION
1	The	DET	2	det
2	admin	NOUN	4	nsubj
3	can	VERB	4	aux
4	modify	VERB	0	ROOT
5	personal	ADJ	6	amod
6	data	NOUN	4	dobj
7	and	CONJ	6	cc
8	the	DET	9	det
9	status	NOUN	6	conj
10	of	ADP	9	prep
11	a	DET	12	det
12	user	NOUN	10	pobj
13	.	PUNCT	0	punct

ID	TEXT	WORD CLASS	DEPENDENCY NUMBER	GRAMMATICAL FUNCTION
1	The	DET	2	det
2	admin	NOUN	7	nsubj
3	and	CONJ	2	cc
4	the	DET	5	det
5	user	NOUN	2	conj
6	can	VERB	7	aux
7	modify	VERB	0	ROOT
8	personal	ADJ	9	amod
9	data	NOUN	7	dobj
10	of	ADP	9	prep
11	that	DET	12	det
12	user	NOUN	10	pobj
13	.	PUNCT	0	punct

Fig. 4. The output of SyntaxNet for the old version (left) and the new version (right) of the sentence in the changed requirement

Analyzing Changes at Word Level. After identifying sentences that have been added, deleted and modified, we proceed to analyze what changes were applied to modified sentences. When a sentence has been modified, we identify word classes in the sentence and for each of these classes, we identify their change type. For identifying word classes we use Google's implementation of a globally normalized transition-based neural network model, called SyntaxNet [12]. SyntaxNet determines the word class (e.g., noun, verb) and the grammatical function (e.g., subject, object) for each word in a sentence. SyntaxNet also identifies dependencies between words and represents them with dependency numbers. We use these later when generating guidance (see Sect. 2.2). Figure 4 shows an example of the output of SyntaxNet.

In order to identify whether words have been added, deleted or modified, we adapted the algorithm implemented in a text-based diff engine, called Text_Diff [13]. Text_Diff detects changes at a phrase level. We process the output from Text_Diff so that we get the changes on a word level.

In the modified sentence from our example: "The admin and the user can modify personal data ~~and the status~~ of that ~~a~~ user", the original Text_Diff algorithm will detect the addition of the phrases "and the user" and "that" and the deletion of the phrases "and the status" and "a". We adapted the algorithm so that it detects additions and deletions of each word in these phrases, as presented in Fig. 5.

The admin <add>and the user</add> can modify personal data ~~and the status~~ of <add>that</add> ~~a~~ user.

The admin <add>and</add> <add>the</add> <add>user</add> can modify personal data ~~and~~ ~~the~~ ~~status~~ of <add>that</add> ~~a~~ user.

Fig. 5. The original (left) and adapted (right) output of Text_Diff

Classify Identified Changes into Relevant and Irrelevant Changes. We consider a change to be relevant if it is likely to impact acceptance tests. Since acceptance tests contain a list of actions to be performed and as actions are generally expressed using verbs in English sentences, we consider verbs as the principal element of analysis in GuideGen. More concretely, we consider a change in a requirement to be relevant if it involves an addition, deletion or modification of a verb or of another word class that relates to a verb such as nouns and adjectives.

If a whole sentence has been added, it is considered to be relevant only if it contains at least one verb. Changes of determiners, adverbs and prepositions are not taken into consideration, since we assume that they do not influence any actions and, therefore, do not have an impact on acceptance tests.

In our example, the following change patterns are considered to be relevant: (1) deletion of the sentence "The addition of a new user must be first approved by the admin", (2) addition of the noun "user", (3) deletion of the noun "status", (4) addition of the sentence "Only user can modify its status" and (5) addition of the sentence "The admin must be logged-in in order to modify personal data of a user". Only these changes are processed in the next steps.

2.2 Generating Guidance

The goal of this step is to generate suggestions about how to modify the affected acceptance tests so that they stay aligned with the changed requirements. An example of a suggestion is *Add new steps or modify existing steps to verify that only user can modify its status*. Every suggestion contains static and dynamic parts.

The static parts of a suggestion differ according to the change patterns identified in the previous step. For instance, if a whole sentence has been added to a requirement, the static part of the suggestion is "Add new steps or modify existing steps to verify that". Accordingly, if a whole sentence has been deleted, the static part of the suggestion is "Delete the steps or their parts which verify that". If a sentence has been modified, the static parts are formulated according to the modification type: whether a verb, subject, object or adjective is added/deleted/modified or a noun is changed from singular to plural, etc.

Table 1. Words included in the dynamic part of a suggestion according to the changed element.

Changed element:	Sentence	Noun		Verb	Adjective
		Subject/conjunction	Object/conjunction		
Words included in the dynamic part:	Changed element (all words in that sentence)	Changed element with adjectives and determiners, related verb and all words that appear after that verb	Changed element, subjects with determiners and adjectives, verbs, prepositions with their objects	Changed element, subjects and objects with determiners and adjectives, prepositions with objects, adverbs	Changed element, related nouns

For instance, if a subject is added, the static parts of the suggestion are "Make sure that now +{*dynamic part*}" and "Add the steps which verify this activity".

The dynamic parts of a suggestion fill the gaps between the static parts. They differ according to the type of the changed element, as shown in Table 1. We defined the rules governing the dynamic parts with informal experimentation and by considering typical sentence structures in requirements documents.

If a whole sentence has been added or deleted, the dynamic part contains all words in that sentence. When a changed element is a subject, the dynamic part contains that subject with its determiners and adjectives, the first related verb and all the words that appear after that verb. We use the word index (ID in Fig. 4) to identify the position of the words. In our example, the following guidance is generated for the added subject "user": "Make sure that now the user can modify personal data of that user."

When the changed element is an object, a verb or an adjective, then the dynamic part contains that element plus its related words. We identify the related words by analyzing word classes, grammatical functions and dependency numbers of words in the modified sentence. Related words for an object are (1) a verb whose index corresponds to the dependency number of the object, (2) a subject whose dependency number refers to the index of the identified related verb and (3) prepositions whose dependency numbers refer to the changed object. We recursively include their related words in the dynamic part. Related words for verbs are (1) directly related subjects, (2) objects, (3) prepositions and (4) adverbs with their related words and corresponding indexes and dependency numbers, while related words for adjectives are the nouns that this adjective directly relates to.

If a subject/object is related to another, main subject/object by a conjunction, we identify the words that are related to the main subject/object. In our example, the deleted object "status" has a conjunction to the direct object "data" (see Fig. 4). Since the verb "modify" with its auxiliary verb "can" is directly related to the object "data", we consider them to be also related to

Table 2. The identified relevant change patterns with the corresponding guidance.

Relevant change patterns	Generated guidance
Change 1: deletion of the sentence "The addition of a new user must be first approved by the admin"	**Delete steps or their parts which verify that** *the addition of a new user must be first approved by the admin*
Change 4: addition of the subject "user"	**Make sure that now** *the user can modify personal data of that user.* **Add the steps which verify this activity**
Change 7: deletion of the object "status"	**Delete steps or their parts which verify that** *the admin can modify the status of a user*
Change 10: addition of the sentence "Only user can modify its status"	**Add new steps or modify existing steps to verify that** *only user can modify its status*
Change 11: addition of the sentence "The admin must be logged-in in order to modify personal data of a user"	**Add new steps or modify existing steps to verify that** *the admin must be logged-in in order to modify personal data of a user*

"status". The subject "admin" refers to the verb "modify" and has a related determiner "the", so they are both classified as related words of the deleted object. The preposition "of" directly refers to "status" and it has the related noun "user" with its determiner "a". The determiner "the" is directly related to "status". The words are ordered by the word index and the dynamic part is formulated as *the admin can modify the status of a user.*

Table 2 presents the guidance that is generated in our example. The guidance consists of one suggestion per change. Static parts are in boldface, while dynamic parts are italicized.

2.3 Notifying Subscribed Parties

In order to ease the communication of changes, we have implemented a notification mechanism that allows requirements engineers to send an automatically generated message to subscribed parties (in particular, testers) when a requirement has been changed. The message contains the summarized changes and the generated guidance. An example is given in Fig. 7 in the next section.

If requirements engineers consider a generated suggestion to be irrelevant, they can mark it so that the tool does not include it in the message. For example, if we add a new sentence: "This should be communicated to Tom.", then the generated suggestion "Add new steps or modify existing steps which verify that this should be communicated to Tom." is irrelevant and can be ignored.

3 Tool Support

We have implemented our approach in a *prototype tool*, a Java web application. The GuideGen tool allows users to upload the list of requirements from an external Excel file, make changes to each of them and notify subscribers (developers, testers, ...) about the changes and the guidance on how to modify the affected acceptance tests.

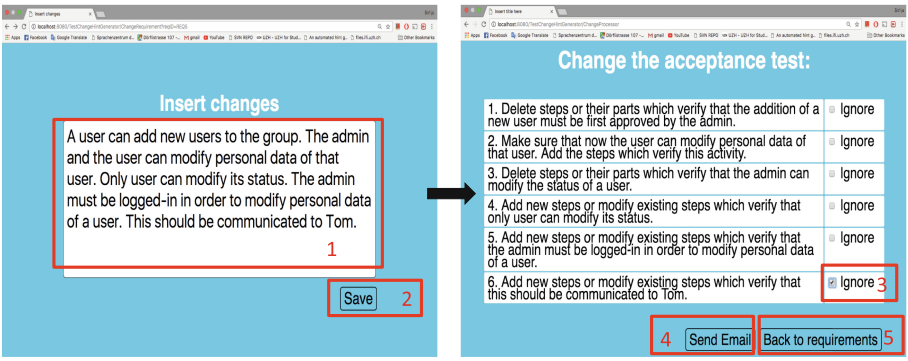

Fig. 6. User interface (UI) of the tool with highlighted process steps.

Figure 6 illustrates the steps taken when using the tool. The left screenshot shows how a user (typically a requirements engineer) can enter changes to a previously selected requirement (step 1) and save them (step 2). Within three seconds, the tool generates guidance consisting of a suggestion for each change and shows it to the user (right screenshot). Suggestions that the user considers to be irrelevant can easily be ignored (step 3). The result can be sent to the subscribed parties in an e-mail generated by the tool (step 4). The user can return to the list of requirements (step 5). Figure 7 shows the e-mail generated by the tool for the given example.

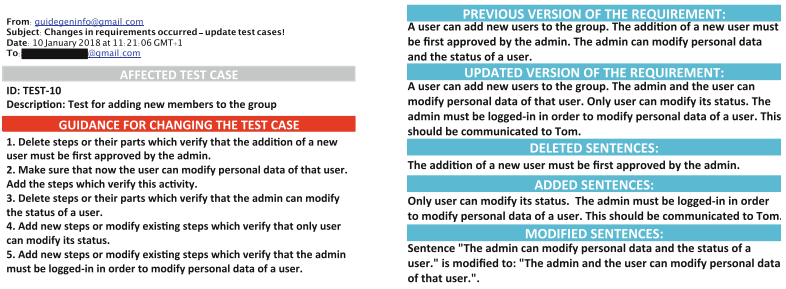

Fig. 7. The e-mail message generated for the example given in Fig. 6.

4 Evaluation

We evaluated GuideGen by applying it to real-world data sets with requirements changes provided by three companies. After pruning the data sets, we ran our tool with the requirements changes contained in the data sets and generated guidance for how to change the associated acceptance tests. The quality of the generated guidance was then assessed by experts from the three companies.

4.1 Study Design

Data collection and analysis. We obtained data sets containing information about changes of requirements from three companies (Table 3). For our evaluation, we needed data records containing the old and the changed version of a requirement and the associated acceptance tests. Table 4 characterizes the data sets.

We pruned the received data sets as follows: (1) We omitted all requirements that had not been changed at all or did not have acceptance tests associated with them. (2) We removed irrelevant changes such as added or deleted punctuation marks, spaces or empty lines. The pruning yielded a total of 448 changed requirements. Our tool filters out semantically irrelevant changes such as addition or deletion of determiners or corrections of typos. On the other hand, for several requirements there was more than one change. So we eventually could evaluate a total of 262 changes (28 for C1, 37 for C2 and 197 for C3).

Table 3. Characteristics of the companies that provided us data sets from one of their projects.

Company	Domain of activity	Software process model	# of employees in total	# of employees on the project	Country
C1	Access control and security solutions	Agile (Scrum)	≈16000	≈120	Switzerland
C2	IT integration, cloud services	Agile (Scrum)	≈500	≈100	Serbia/Germany
C3	Automation for warehouses and distribution centers	Waterfall	≈2500	≈500	Switzerland

Table 4. Characteristics of the data sets used in our evaluation study.

Company/Data set	Type of requirements	# of requirements in the data set	# of considered requirements	# of evaluated changes
C1/DS1	User story	157	20	28
C2/DS2	User story	30	30	37
C3/DS3	Classic textual requirement	5301	398	197

Running the tool. For every of the 262 evaluated changes, we generated guidance for how to change the associated acceptance tests using our tool prototype. We uploaded the old version of the requirements into the tool, replaced each of them with the new version, and recorded the generated guidance.

Assessing the quality of the generated guidance. The generated guidance was assessed by experts from the three companies. An overview of the experts and their experience is provided in Table 5.

95 changes were fully assessed by two or three experts. We created a questionnaire[2] in which, for every requirement, we presented the old and the changed requirement, the associated acceptance tests and the guidance for changing the acceptance tests generated by our tool. For each suggestion provided in the guidance, we asked six questions to assess the quality of the suggestion: (1) Is the suggestion correct in terms of actions that need to be performed? (2) Is it grammatically correct? (3) Is it complete? (4) Does the expert understand what has been suggested by the tool? (5) Would the expert be able to perform an update of the impacted acceptance test without any further clarifications? (6) Is the suggestion redundant or unnecessary? Finally, we asked whether there is anything missing from the guidance for a changed requirement (i.e., from the set of all suggestions generated for that requirement). Questions 1–3 and 5 had to

[2] https://docs.google.com/forms/d/1vLJYFIjmtLjzC60e2iT3JLbs9ST8LmOOhO9kotfrBwo/edit. For confidentiality reasons, the file does not contain the real data from our data sets, but only the example shown in this paper.

Table 5. Characteristics of the experts who participated in the study.

Company	Participant	The role of participant	Years of experience in IT	Years on the current position
C1	P1	Requirements engineer	10	4
C1	P2	Senior test analyst	12	4
C2	P3	Requirements engineer	6	3
C2	P4	Senior test engineer	7	4
C3	P5	Requirements engineer	10	5
C3	P6	QA manager	12	6
C3	P7	Test engineer	4	4

be answered on a five-point Likert scale (from "strongly disagree" to "strongly agree"). In case of non-agreement, the expert was asked to provide an explaining text. Question 4 was a yes/no question, while Question 6 and the final question about missing suggestions were answered as free text.

In company C3, due to limited availability of the experts, only 30 suggestions could be thoroughly assessed by all three experts. The suggestions generated for the remaining 167 changes could only be assessed for correctness by a single expert.

When the experts had finished answering the questionnaire for all changed requirements assigned to them, we conducted a short interview where we asked them seven questions about the usefulness and applicability of our approach[3].

4.2 Results

In this sub-section we present the results of the assessment of the generated guidance by the experts and some key insights from the follow-up interviews.

All 262 changes were correctly identified in terms of the change type, showing that the algorithm for identifying added, deleted and modified sentences with a modification threshold of 0.6 performs accurately. Table 6 presents the results of the evaluation of the guidance generated for 95 changes in requirements by the experts.

For calculating the percentages in Table 6 for the questions answered on a Likert scale, we interpreted the values 4 ("Agree") and 5 ("Strongly agree") as "yes". Analogously, we interpreted 1 ("Strongly disagree") and 2 ("Disagree") as "no". 3 ("Neutral") was interpreted according to the textual explanation provided by the experts. From eleven such answers three were interpreted as "yes" and eight as "no".

Table 6 shows that in C1 and C2 the experts assessed more than 80% of the suggestions as correct in terms of actions. In C3 one expert was more negative

[3] https://docs.google.com/forms/d/1rk-P-m4sd8rpHk_umForPW6QebWRnoLBjfexhBqiVI4/edit.

Table 6. The quality of the generated suggestions based on an assessment by industrial experts.

Generated in total/ assessed	Company/ Partici- pant	Correct in terms of actions	Gramma- tically correct	Complete	Under- standable	Self- expla- natory	Redundant/ unnece- ssary	Missed changes
28/28	C1/P1	89.2%	82.1%	100%	100%	75%	7.1%	3.6%
	C1/P2	89.2%	82.1%	100%	100%	75%	7.1%	3.6%
37/37	C2/P3	81%	67.5%	94.6%	100%	75.6%	10%	5.4%
	C2/P4	81%	67.5%	94.6%	100%	75.6%	10%	5.4%
197/30	C3/P5	50%	86.6%	96.6%	93.3%	70%	50%	3.3%
	C3/P6	70%	80%	93.3%	100%	73.3%	30%	3.3%
	C3/P7	66.7%	86.6%	96.6%	93.3%	73.3%	33.3%	3.3%

than the other two, especially regarding the correctness in terms of actions. This is due to a misunderstanding: expert P5 classified all redundant suggestions as wrong in terms of actions, i.e., when they were actually correct, but unnecessary. So we can consider the correctness of our guidance for data set 3 to be at least 66.7%.

"- The section 3 contains:
– Doctors' corner
– Register your practice opens a form inline or a popup with:
— Name of your practice (mandatory)
— Contact phone (mandatory)
— Contact e-mail (mandatory)
— Give us your contact details and we will get back to you soon! "

Fig. 8. Example of a changed requirement from C2. Added text is in green and underlined.

Figure 8 shows a change (in the acceptance criteria of a user story) where GuideGen does not work such well. According to the experts, the text means that Sect. 3 of a web page contains a label "Doctors' corner" and a button "Register your practice". When a user clicks on the button, a pop-up window is displayed. The change in the requirement is that an additional message shall be displayed in this window.

For this change, the GuideGen tool generated the following suggestion, which the experts considered to be wrong both in terms of actions and grammatically: "Add new steps or modify existing steps which verify that the Sect. 3 contains register your practice opens a form inline or a pop-up with give us your contact details and we will get back to you soon!". This result may indicate that our approach does not perform well on ill-structured texts (the experts confirmed that this text is not formulated well). However, it may also indicate that our treatment of enumerations (cf. sentence level analysis in Sect. 2.1) needs improvement.

The last column in Table 6 presents the number of changes that were relevant, but not detected by GuideGen. In C1 a noun without any related verbs was added. This was classified as an irrelevant change and hence no guidance was generated. Further, the current version of our tool does not consider the change of numerical values as a relevant change pattern. Hence, no guidance was generated for two such cases in C2 and one in C3. This problem will be fixed in the next release of the tool.

As stated above, the guidance for 167 changes in requirements from company C3 could not be evaluated fully due to limited availability of the experts. Table 7 shows the results of the assessment of the generated suggestions for these changes.

Table 7. Suggestions assessed for correctness in terms of actions by a single expert only.

Company/ Participant(role)	Assessed suggestions	Correct in terms of actions	Wrong (rephrasing only)	Wrong (only clarifications or notes added or deleted)	Wrong (due to tool limitations)
C3/P6 (QA)	167	70.6%	10.2%	13.8%	5.4%

We found that 70.6% were correct in terms of actions, while 24% were incorrect because the changes only rephrased a requirement or added or deleted only clarifications or notes. A small percentage (5.4%) of wrong suggestions were due to limitations of our prototype tool (e.g., wrongly identified dependencies).

Next, we present the main findings from the follow-up interviews with the experts regarding the overall usability and usefulness of GuideGen. All experts stated that GuideGen can be helpful in communicating changes on time and with less effort, it can help test engineers to make a decision on how to update acceptance tests and they would be willing to slightly adapt their style of writing requirements in order to ensure better quality of guidance. Four experts emphasized that one of the reasons for wrongly generated guidance was the poor quality of the requirements. They stated that suggestions can be too general, but that this is directly related to the level of detail specified in the requirements. The experts from C1 stated that the approach would be even more useful if it could highlight the parts of the acceptance tests that should be changed directly in the acceptance test document. With respect to the usability of the tool, P1 and P2 suggested an improvement of the user interface so that the tool navigates directly to the steps that are suggested to be changed.

4.3 Threats to Validity

Internal and construct validity. Our evaluation strongly depends on the expertise of the people who assessed the guidance generated by GuideGen. In order to foster validity, we aimed at assessing each guidance by at least two experts. In company C3, due to limited availability of experts, we could assess

only 30 cases this way, while the rest was evaluated only in terms of correctness by a single expert. We tried to mitigate this problem by including all types of changes in the fully evaluated sample from company C3. Even with this restriction, the workload for the experts was high, since they needed to answer six questions per 28 and more suggestions, which might impact the quality of their answers. Therefore, we provided an online access to the questionnaire, so that the experts could answer the questions in iterations.

External validity. The generalizability of our results is limited by the fact that our evaluation covers data sets from only three companies. We tried to improve generalizability by including both agile and traditional requirements artifacts as well as different types of changes in our data sets. Although the study involves only seven participants, we had at least two participants per data set and we tried to keep diversity in terms of roles, so that requirements engineers and test managers are included.

5 Discussion

The results presented in Table 6 show that the quality of the generated guidance differs from company to company. This is not surprising as the outcome of our natural language processing techniques depends on the type and quality of requirements artifacts and on the content that is being changed in these artifacts.

GuideGen performs better for user stories than for traditional requirements. This is probably due to the fact that user stories typically are more concise and describe features more precisely than traditional requirements do. Further, text changes in traditional requirements documents often do not bring any novelty to the feature that is being described, but only provide clarifications or simply rephrase the text.

The complexity of a sentence also affects the quality of the guidance generated. On the one hand, very short or incomplete sentences affect both the correctness and completeness of suggestions and may even cause the omission of relevant changes. On the other hand, long, complex sentences which contain one or more relative clauses or statements in parentheses may cause problems: word classes, their grammatical functions and dependencies between words in a sentence may be wrongly identified, which leads to wrongly generated guidance.

Our approach currently cannot recognize certain types of irrelevant changes, for example, when mere comments such as "This should be communicated to Tom" are added. Wrong suggestions are generated in this case. However, our tool allows a requirements engineer to remove such false positives easily before communicating changes and generated guidance to subscribers (cf. Fig. 7).

GuideGen needs only sets of old and changed requirements (and their associated acceptance tests) as input. This is both a strength and a limitation. It is a *strength* because with our tool, requirements engineers can easily communicate requirements changes together with guidance on how to change the acceptance

tests that correspond to the changed requirements. On the other hand, it is a *limitation*, as our tool does not analyze which artifacts are impacted by a changed requirement. This problem is addressed by research on automated traceability and change impact analysis [4,8,14].

6 Related Work

Many researchers investigate requirements traceability for supporting change impact analysis. For example, Antoniol et al. [15], Marcus and Maletic [14], De Lucia et al. [4] and Hayes et al. [16] use information retrieval methods to ensure automated traceability for change impact analyses. Others employ natural language processing. For example, Arora et al. [8] analyze the impact of changes in a requirement on other requirements in a system using NLP methods. However, all these approaches focus on identifying which requirements or other artifacts are impacted by a change in a requirement, while we investigate how to manage the change and which actions to perform in order to keep requirements and acceptance tests aligned.

Bridging the communication gap among people involved in developing a system draws attention of researchers and practitioners. Sinha et al. [17] define and explain the communication problems when managing requirements in a distributed environment. Bjarnason and Sharp [18] and Adzic [19] emphasize the communication problems between requirements engineers, developers and testers in agile projects. By generating guidance in natural language that can be easily communicated to the interested parties via e-mail, our approach supports easy and timely communication of changes between requirements engineers and developers/testers.

7 Conclusions

Summary. We presented GuideGen, a tool-supported method for automatically generating guidance on how to align acceptance tests with evolving requirements. With a correctness score of more than 80% for real-world agile requirements and around 67% for traditional requirements, our approach provides useful guidance for maintaining acceptance tests and keeping them aligned with the evolving requirements.

Future Work. We will improve GuideGen based on the evaluation results and then perform a more thorough evaluation of its overall usefulness and usability.

Acknowledgements. We thank our experts and their companies for investing time and effort into the evaluation of our approach. This work was partially funded by the Swiss National Science Foundation under grant 200021-157004/1.

References

1. Bjarnason, E., Runeson, P., Borg, M., Unterkalmsteiner, M., Engström, E., Regnell, B., Sabaliauskaite, G., Loconsole, A., Gorschek, T., Feldt, R.: Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empir. Softw. Eng. **19**(6), 1809–1855 (2014)
2. Hotomski, S., Ben Charrada, E., Glinz, M.: An exploratory study on handling requirements and acceptance test documentation in industry. In: 24th IEEE International Requirements Engineering Conference (RE 2016), pp. 116–129. IEEE (2016)
3. Borg, M., Gotel, O.C., Wnuk, K.: Enabling traceability reuse for impact analyses: a feasibility study in a safety context. In: 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE), pp. 72–78. IEEE (2013)
4. De Lucia, A., Marcus, A., Oliveto, R., Poshyvanyk, D.: Information retrieval methods for automated traceability recovery. In: Cleland-Huang, J., Gotel, O., Zisman, A. (eds.) Software and Systems Traceability, pp. 71–98. Springer, London (2012). https://doi.org/10.1007/978-1-4471-2239-5_4
5. Nair, S., de la Vara, J.L., Sen, S.: A review of traceability research at the requirements engineering conference$^{re@21}$. In: 21st IEEE International Requirements Engineering Conference (RE 2013), pp. 222–229. IEEE (2013)
6. Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. Wiley, New York (1997)
7. Myers, G.J., Sandler, C., Badgett, T.: The Art of Software Testing. Wiley, New York (2011)
8. 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 (RE 2015), pp. 6–15. IEEE (2015)
9. Hotomski, S., Ben Charrada, E., Glinz, M.: Aligning requirements and acceptance tests via automatically generated guidance. In: 4th Workshop on Requirements Engineering and Testing (RET) (2017)
10. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: ACL (System Demonstrations), pp. 55–60 (2014)
11. Rus, V., Lintean, M.C., Banjade, R., Niraula, N.B., Stefanescu, D.: Semilar: the semantic similarity toolkit. In: ACL (Conference System Demonstrations), pp. 163–168 (2013)
12. Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., Collins, M.: Globally normalized transition-based neural networks. arXiv preprint arXiv:1603.06042 (2016)
13. Hagenbuch, J.S.C.: Text_diff-engine for performing and rendering text diffs. https://pear.horde.org/
14. Marcus, A., Maletic, J.I., Sergeyev, A.: Recovery of traceability links between software documentation and source code. Int. J. Softw. Eng. Knowl. Eng. **15**(05), 811–836 (2005)
15. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E.: Recovering traceability links between code and documentation. IEEE Trans. Softw. Eng. **28**(10), 970–983 (2002)
16. Hayes, J.H., Dekhtyar, A., Sundaram, S.K.: Advancing candidate link generation for requirements tracing: the study of methods. IEEE Trans. Softw. Eng. **32**(1), 4–19 (2006)

17. Sinha, V., Sengupta, B., Chandra, S.: Enabling collaboration in distributed requirements management. IEEE Softw. **23**(5), 52–61 (2006)
18. Bjarnason, E., Sharp, H.: The role of distances in requirements communication: a case study. Requir. Eng. **22**(1), 1–26 (2017)
19. Adzic, G.: Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing. Neuri Limited, London (2009)